Proyecto Final de Carrera
Ingeniería en Informática
Curso 2010/2011

Interpretación de carteles con la cámara de un móvil

Ana Belén Cambra Linés

Abril de 2011

Directora: Ana Cristina Murrillo Arnal

Departamento de Informática e Ingeniería de Sistemas
Centro Politécnico Superior
Universidad de Zaragoza

RESUMEN

Mucha de la información que recibimos es visual y, cada vez encontramos más cámaras y bases de datos de imágenes a nuestra disposición. Por ello, el procesamiento automático e "inteligente" de imágenes tiene mucho interés en el desarrollo de nuevas tecnologías y aplicaciones basadas en visión artificial. En particular, en este proyecto el trabajo se centra en las tecnologías en auge de aplicaciones móviles, y cómo hacer uso de las cámaras integradas en los *smartphones* y de su capacidad cada vez mayor de cómputo. Gracias a esto, se pueden desarrollar aplicaciones relacionadas con la visión por computador en móviles, algo impensable hasta hace poco debido a las grandes limitaciones que presentaban.

En el presente proyecto se desarrolla una aplicación para el *iPhone* capaz de extraer el texto de carteles rectangulares presentes en una imagen. Aunque actualmente existen muchos reconocedores de caracteres, llamados Optical Character Recognitions (OCRs), que permiten extraer el texto de una imagen, sus buenos resultados están muy condicionados a cómo se presenta el texto dentro de dicha imagen. Se requiere que el usuario enfoque con mucha precisión dónde se encuentran los textos a leer. Esta situación es una gran restricción y sobretodo muy poco realista y robusta, además de no permitir aprovechar estas tecnologías para, por ejemplo, dar servicios a personas con problemas de visión. Por ello, un objetivo principal de este proyecto es desarrollar una aplicación que libere al usuario de tal restricción.

El funcionamiento de la aplicación desarrollada puede resumirse en tres pasos: elección, procesamiento y lectura del texto de una imagen. Primero el usuario debe capturar una imagen. En el segundo paso se procesa dicha imagen para obtener una nueva que sea más adecuada, para que en el último paso, su texto pueda ser extraído fácilmente por un OCR ya existente integrado también en el teléfono. El trabajo desarrollado en este proyecto, se centra sobretodo en el segundo paso: diseñar e implementar un proceso por el cual obtener una imagen adecuada para conseguir unos buenos resultados con un OCR, y en diseñar un prototipo que presente un funcionamiento satisfactorio en el teléfono.

Para ello, antes de comenzar con la fase de desarrollo ha sido necesario una familiarización con el entorno: desde el sistema operativo al entorno de programación, así como estudiar la viabilidad de la inclusión de librerías estándar al dispositivo. En el proyecto se ha diseñado e implementado un detector de rectángulos y un modelo para evaluar la probabilidad de que éstos contengan texto. También se han comparado tres OCRs con el fin de seleccionar aquel que mejor se adapta al proyecto y se ha integrado todo lo anterior creando un prototipo real para el *iPhone*. La aplicación se ha probado tanto en el simulador como en dos dispositivos físicos: un *iPhone* 4 y un *iPod Touch*. Los resultados obtenidos han sido satisfactorios, consiguiendo un prototipo realista, y que podría utilizarse tanto como traductor de textos como asistente de lectura ante deficiencias visuales.

Agradecimientos

Mi primer agradecimiento es para ti, Ana Cris, gracias por guiarme a lo largo de todo este proyecto, por los ánimos recibidos en momentos de crisis y sobretodo por tu interés. Me ha encantado que hayas sido la directora de mi proyecto.

En especial a "caramelito" y "cuchurita", porque gracias a "ARToolKit" descubrí con vosotras que hay vida en el CPS pasadas las 5 de la madrugada... Gracias chicas, por vuestros ánimos y paciencia.

A ti hippie, por estar allí siempre (aún estando a 2000 Km), por tu apoyo y paciencia pero sobretodo por lo mucho que me quieres y como soy cuando estoy contigo.

A vosotras crías, por todos los momentos vividos juntas, por vuestra ayuda, cariño y comprensión en momentos de desesperación. También a vosotros, papá y mamá, por la paciencia demostrada con mi carácter y mal humor, por el apoyo incondicional que me dais y por estar siempre ahí, por y para nosotras. También al resto de mi familia porque estar siempre para lo bueno y lo malo.

A Naiara, por estar conmigo, siempre dispuesta a escucharme y entenderme.

También gracias a todos los que me habéis escuchado y aguantado mis interminables conversaciones sobre imágenes, rectas, esquinas, histogramas y OCRs... En definitiva, gracias a todos los que formáis parte en mi vida.

"Sólo hay 10 tipo de personas en el mundo, las saben binario y las que no"

Índice general

Capítulo 1

Introducción

Cada día es más habitual hacer un uso diario del teléfono móvil para multitud de tareas, aparte de las típicas de un teléfono común. Además, dicho uso se ve incentivado por la aparición de los móviles de última generación, los llamados *smartphones*. Un *smartphone* es el término comercial que denomina a un teléfono móvil que ofrece algunas de las características que posee un ordenador personal, además de proporcionar las funcionalidades propias de una *PDA*. Existen multitud de modelos fabricados por diferentes empresas, como puede verse en la Figura 1.1, cuyo funcionamiento se basa sobre diferentes sistemas operativos. Dependiendo del fabricante y modelo, los *smartphones* pueden incluir distintos dispositivos y funcionalidades, como por ejemplo: cámara de fotos y/o vídeo de alta definición integrada, acceso a internet vía *Wi-Fi*, *GPS*, acelerómetro, teclado *qwerty* y/o una pantalla táctil; así como distintas aplicaciones: navegador Web, servicios de correo electrónico, editor/lector de textos, etc. Sin embargo, todos comparten una característica destacable: la posibilidad de desarrollar aplicaciones para su instalación en el teléfono.

Figura 1.1: Diferentes modelos de *smartphones*

En este proyecto, se trabaja con la plataforma *iPhone* de *Apple*[1]. Dadas sus características, ofrece multitud de posibilidades a la hora de diseñar y desarrollar nuevas aplicaciones, además de proporcionar la oportunidad de aprender a manejar un nuevo sistema operativo y lenguaje de programación. En particular, intentando aprovechar que contiene una cámara y una buena capacidad de cómputo, se pretende investigar su uso en aplicaciones que realicen tareas relacionadas con la visión por computador.

La visión por computador es una rama de la inteligencia artificial que pretende dotar a un computador de la capacidad de analizar la información del medio que le rodea a partir de imágenes. La tarea a la hora de desarrollar un sistema de visión artificial que imite o sustituya al humano es complicada, ya que la mayoría del procesamiento que realizamos es inconsciente y desconocido, por lo que es difícil de transportar y replicar en un sistema informático. El gran número de cámaras presentes en nuestro entorno permite su uso en diversas aplicaciones, y la convierte en una disciplina de creciente y continuo interés dentro del campo científico-técnico. Algunas de sus aplicaciones están destinadas a la robótica, los procesos de inspección automática, la navegación autónoma de vehículos, el reconocimiento de objetos, el análisis de imágenes médicas...

En este proyecto se pretende desarrollar una aplicación que sea capaz de interpretar carteles del entorno con sólo realizar una fotografía. Dicha aplicación está destinada tanto a un usuario de carácter general, pudiendo ser usada como un traductor, así como a personas con algún tipo de discapacidad visual, pudiendo ser empleada como intérprete o asistente visual.

El trabajo realizado en este proyecto está enmarcado dentro de uno de los proyectos de investigación del grupo de Robótica, Percepción y Tiempo Real del Departamento de Informática e Ingeniería de Sistemas, el proyecto VISPA DPI2009-14664: "non-conventional VIsion System for Personal Assistance".

1.1. Trabajo relacionado

El campo del desarrollo de aplicaciones para dispositivos móviles está creciendo junto con el avance de estas tecnologías. Entre las aplicaciones existentes para *smartphones* relacionadas con la temática de este proyecto se encuentra *Word Lens*[2], desarrollada para el *iPhone* y lanzada al mercado en Diciembre de 2010. Permite traducir instantáneamente palabras impresas de un lenguaje a otro gracias a la cámara de vídeo integrada, en tiempo real y sin necesidad de conectarse a internet.
Con anterioridad, también se encuentran aplicaciones desarrolladas para otros dispositi-

[1] http://www.apple.com/
[2] http://itunes.apple.com/es/app/word-lens/id383463868?mt=8

vos móviles, como es el caso de *Dictionary 6 in 1 with camera function*[3] para la Nintendo DS, capaz de traducir palabras escritas en hasta seis idiomas diferentes a partir de una foto, utilizando también técnicas basadas en el reconocimiento de caracteres. Asimismo, dentro del mismo grupo, se llevó a cabo otro trabajo de investigación [1] que se centró en construir un reconocedor de caracteres de un alfabeto más complejo que el latino, como es el japonés, para su uso en el *iPhone*. El presente proyecto intenta contribuir en mejorar la robustez de este tipo de aplicaciones y analizar las posibilidades en la plataforma elegida.

En el campo de la visión por computador el reconocimiento y la detección de zonas con texto en imágenes así como, el reconocimiento de caracteres [2] son temas en continuo desarrollo y avance. El reconocimiento automático de caracteres, inicialmente, iba dirigido a aplicaciones de digitalización de textos escritos, tipográficos o manuscritos en todo tipo de soportes. Poder automatizar la introducción de caracteres, evitando la entrada por teclado, implica un importante ahorro de recursos humanos y un aumento de la productividad, al mismo tiempo que, se mantiene, o se mejora, la calidad de muchos servicios. En el mercado existen ya aplicaciones destinadas a digitalizar el texto existente en una imagen. Estas aplicaciones se denominan *OCRs* (Reconocimiento Óptico de Caracteres). Sin embargo, su éxito está muy sujeto a las condiciones de la imagen. Si el texto no se encuentra perfectamente segmentado en la imagen, este tipo de aplicaciones son incapaces de detectarlo, más adelante, en la Figura 1.2, se muestra un ejemplo. Desde hace tiempo, se investiga un paso más, intentando encontrar zonas de imágenes arbitrarias que puedan contener texto, para dar la zona segmentada al *OCR*. Esta detección de texto automática se realiza siguiendo distintas estrategias: por ejemplo en [3] se segmenta la imagen en zonas analizando su textura a lo largo de varios pasos y filtros; otros resultados más recientes como en [4], las zonas de texto se reconocen a través de un clasificador, en este caso, una combinación de varios clasificadores *SVM (Support Vector Machine)* [5]. Durante el desarrollo de este proyecto se han estudiado varios trabajos y propuestas para localizar las zonas de texto en imágenes generales, pero la mayoría no eran aplicables directamente por el coste o complejidad [6][7][8][9][10] siendo poco adecuados para su ejecución en el móvil. El estado de investigación de estos temas es muy avanzado. Para evitar la sobrecarga de algunos servicios Web, se desarrollaron los *CAPTCHAs*, que consisten en una prueba desafío-respuesta utilizada en computación para determinar cuándo el usuario es o no humano. La típica prueba consiste en que el usuario introduzca un conjunto de caracteres que se muestran en una imagen distorsionada que aparece en pantalla. Se supone que una máquina no es capaz de comprender e introducir la secuencia de forma correcta por lo que solamente el humano podría hacerlo. Pero existen casos [11] donde ya se han superado algunas de estas pruebas gracias a un procesado avanzado de la imagen.

El trabajo desarrollado en este proyecto está relacionado con el segundo grupo de trabajos, que intentan procesar y segmentar la imagen lo mejor posible para facilitar el

[3] http://dsiware.nintendolife.com/reviews/2009/09/dictionary_6_in_1_with_camera_function_dsiware

trabajo y mejorar los resultados de un *OCR*, pero en este proyecto en particular, con las adaptaciones necesarias para su ejecución en un móvil.

1.2. Motivación

Para ilustrar el entorno de aplicación deseado de este proyecto se muestra el comportamiento de un *OCR* existente ante dos tipos diferentes de imágenes. Las imágenes pueden verse en la Figura 1.2. El *OCR* utilizado es gratuito y está disponible en la red[4] sin necesidad de instalar nada en el ordenador. En la primera imagen, el texto del cartel está totalmente de frente, mientras que en la segunda aparece un cartel que debido a la inclinación y perspectiva, contiene un texto ilegible para el *OCR*. Los resultados de ejecutar el mencionado *OCR* con estas imágenes pueden verse en la misma figura, donde se comprueba que si el encuadre del texto es preciso y frontal, se obtienen un resultado perfecto, mientras que en caso contrario, éste no es capaz de reconocer el texto. Con este simple ejemplo, se observa que los *OCRs* actuales funcionan perfectamente cuando los caracteres a reconocer se encuentran en unas condiciones óptimas.

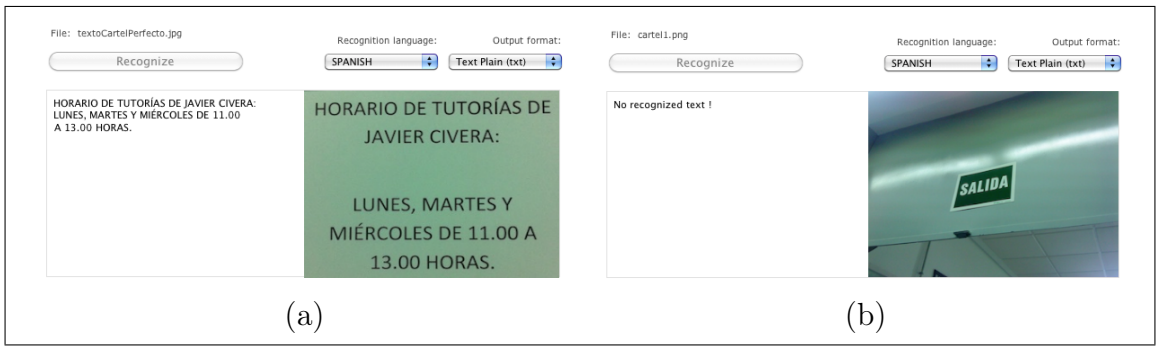


Figura 1.2: (a) Resultados obtenidos por el *OCR* usando una imagen de cerca, frontal y encuadrada y (b) otra desde una mayor distancia, donde el cartel sufre una deformación debido a la perspectiva

Como se ha comentado, este proyecto pretende desarrollar un proceso mediante el cual se obtenga de una imagen cualquiera un segmento adecuado para que un *OCR* reconozca y extraiga su texto. Un punto importante es liberar al usuario de la restricción de que la imagen original contenga un texto de frente y perfectamente encuadrado llenando casi toda la imagen. De esta manera, se puede conseguir desarrollar una aplicación más robusta, que puede ayudar también a personas con algún tipo de discapacidad visual.

[4] *OnlineOCR*: http://www.onlineocr.net/

1.3. Objetivos

En pocas palabras, el objetivo general de este proyecto es interpretar un cartel procedente de una fotografía tomada desde un móvil. El proceso completo se realiza en el propio móvil: desde la captura o elección de la imagen hasta la obtención del texto procedente del cartel, o carteles, existentes en ella.
Al desconocer a priori la ubicación del cartel a procesar, el primer paso es determinar en qué zonas de la imagen es más probable que se encuentre un cartel rectangular, para posteriormente, analizar su contenido en busca de aquellos donde hay mayor probabilidad de encontrar texto. De esta manera, se le proporcionan al *OCR* únicamente aquellas zonas de la imagen que son más probables que contengan texto.

Para conseguirlo, las tareas desarrolladas han sido las siguientes (el detalle temporal de dichas tareas se encuentra en el Anexo B):

- Familiarización con las herramientas necesarias para el desarrollo de este proyecto, detalladas en el Anexo A.
- Estudio de trabajos relacionados existentes.
- Diseño y desarrollo de un proceso por el cuál, a partir de una imagen, se extraigan las zonas que más probablemente correspondan con un cartel. En concreto, es necesario implementar un proceso para extraer las rectas de la imagen, buscar puntos de corte entre ellas, y encontrar esquinas con las que formar hipótesis de rectángulos en la imagen. Además, diseñar un modelo con el que evaluar la probabilidad de que contengan texto, y finalmente realizar un preprocesado en las hipótesis aceptadas para pasarlas a un *OCR* convencional.
- Estudio de *OCRs* disponibles y análisis de su viabilidad para integrarlos en el *iOS*.
- Implementación y evaluación del proceso diseñado integrado en la plataforma *iOS*.
- Documentación y análisis de resultados.

1.4. Estructura de la memoria

En la presente memoria se describe el proceso llevado a cabo para el desarrollo de un intérprete de carteles en un dispositivo móvil. Su contenido se ha distribuido en cuatro capítulos incluyendo esta introducción. En el capítulo 2 se presenta todo el proceso teórico necesario para el desarrollo de la aplicación. En el capítulo 3 se muestra el diseño de la aplicación, así como una comparación entre la aplicación desarrollada y el OCR con objeto de evaluar la contribución del trabajo llevado a cabo. En el capítulo 4, las conclusiones recopilan los resultados del proyecto y el trabajo futuro, así como la valoración personal del trabajo realizado. Finalmente, varios anexos amplían detalles de las distintas partes.

Capítulo 2

Proceso de desarrollo de un intérprete de carteles

Como ya se ha comentado brevemente en los objetivos, se ha diseñado un proceso con los pasos necesarios para construir una aplicación que sea capaz de extraer el texto de una imagen, como se muestra en la Figura 2.1.

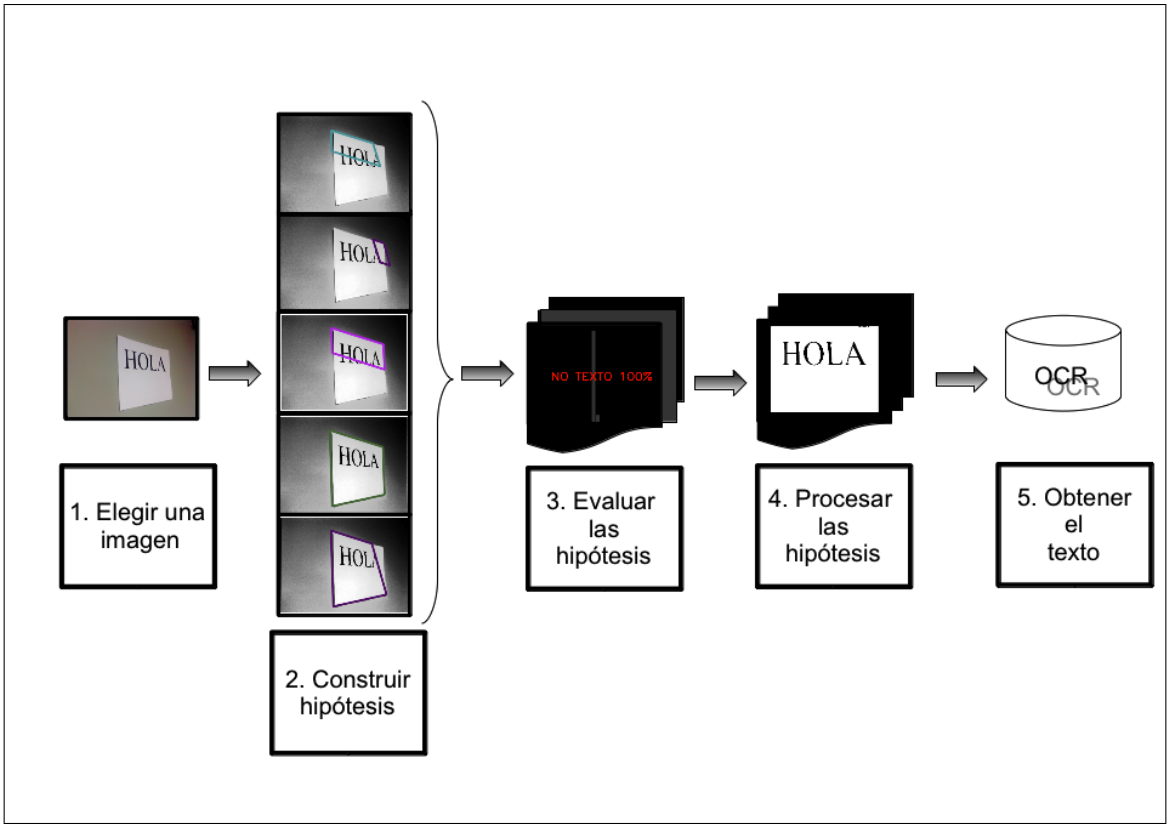


Figura 2.1: Esquema simplificado del proceso diseñado

A lo largo de este capítulo, se profundiza en cada paso realizado durante el proceso, centrando especial atención en los conceptos teóricos necesarios para su comprensión y posterior implementación.

2.1. Búsqueda de rectángulos

Uno de los objetivos ya mencionado de este proyecto es liberar al usuario de la imposición de que la imagen a leer, contenga un texto de frente y perfectamente encuadrado. Por lo tanto, al desconocer a priori la ubicación del cartel a procesar, y poder ser ésta cualquiera dentro de la imagen, el primer paso es buscar su posible ubicación en ella.
Mirando a nuestro alrededor se observan carteles de muy diferentes colores, tamaños y formas, aunque como la mayoría se asemeja a una forma rectangular, el trabajo se va a centrar en este tipo. Por lo tanto, el primer paso es detectar formas rectangulares.

Figura 2.2: Diferentes carteles con texto

El proceso implementado para detectar los rectángulos, inspirado en [12], consiste en obtener rectas que aparecen en la imagen, para a partir de ellas encontrar esquinas y así, construir hipótesis de posibles rectángulos intentando agruparlas. Con este proceso, se consigue reducir la zona a analizar de la imagen, intentando procesar sólo aquella que puede ser de interés.

2.1.1. Preprocesado de la imagen

En todo proceso de visión artificial el preprocesamiento de una imagen tiene como finalidad producir otra de mayor calidad que simplifique y facilite etapas posteriores. El objetivo es actuar convenientemente sobre los niveles de gris de la imagen para compensar posibles defectos de iluminación así como eliminar ruido y efectos espurios.

Técnicas de realce. Su objetivo principal es aumentar el contraste de una imagen mediante la redistribución de sus niveles de gris. En visión por computador, la finalidad de este aumento de contraste no es mejorar la calidad de la imagen para una mejor visualización, sino obtener una imagen resultante que sea más adecuada que la original

para una determinada aplicación. Dentro de estas técnicas se encuentra la equalización del histograma [13] que reasigna los niveles de gris de la imagen original para que queden más uniformemente repartidos.

Técnicas de suavizado. Su objetivo principal es reducir el ruido y/o espurios que pueden presentarse en una imagen. El ruido no es más que información no deseada que contamina la imagen, es decir, píxeles representados en la imagen que no existen en realidad en la escena capturada. Este ruido es adherido a la imagen a causa del proceso de captura, de su digitalización o incluso de su transmisión. Estas técnicas tienen en cuenta el nivel de gris de los píxeles vecinos para corregir el valor de cada píxel [14]. Las diferentes variantes difieren en cómo escogen los píxeles vecinos y en la ponderación que se les da a cada uno de ellos. En cualquier caso, esta operación se lleva a cabo mediante la convolución de la imagen con una máscara. Y aunque estas técnicas consigan gratamente reducir el ruido existente, tienen un gran inconveniente: el enturbiamiento que se produce en la imagen, provocando el difuminado de los bordes, por lo que deben ser usadas con cierta precaución.

2.1.2. Detección de rectas

El primer paso para poder encontrar las rectas presentes en una imagen es extraer los contornos de la imagen. Son regiones de la imagen donde existe un cambio brusco en el nivel de gris entre los píxeles adyacentes y normalmente su causa principal es la intersección de los objetos. Además de suministrar una valiosa información reducen de manera considerable la cantidad de información a manejar; eliminando información innecesaria pero preservando las características fundamentales de la imagen. En el Anexo C se encuentra información más detallada sobre la extracción de contornos.

Una vez extraídos los píxeles de contorno, se desea agrupar los segmentos en rectas. Existen distintos métodos como son: la transformada de Hough [13] o el método de Burns [15]. Aquí se ha utilizado el método [16], ya que tras unos experimentos iniciales proporciona mejores resultados que los nombrados anteriormente en el tipo de imágenes que se utilizan (en el Anexo C.1 puede verse un ejemplo de los resultados obtenidos con cada uno de ellos). En este método, las rectas se obtienen agrupando los píxeles mediante un algoritmo de seguimiento de contornos teniendo en cuenta la orientación del gradiente de cada píxel así como la orientación de sus píxeles vecinos. Todas las posibles orientaciones de la dirección del gradiente se dividen en ocho conjuntos. Los píxeles de contorno pertenecientes a un mismo conjunto se unen en cadenas siguiendo un algoritmo de seguimiento de contornos. Este algoritmo los agrupa mientras que sean píxeles vecinos, considerando píxeles vecinos a los ocho adyacentes. El hecho de que los píxeles pertenecientes a un segmento de recta caigan en distintos conjuntos, por su dirección del gradiente, se ven

solventadas en la etapa de seguimiento de contornos, donde se tiene en cuenta también los píxeles de los conjuntos adyacentes.

Filtrado de rectas. Como resultado de la extracción se obtienen segmentos de rectas representadas por su punto inicial y su punto final. Para evitar tener segmentos de rectas repetidos, que debido al ruido o a la discretización caracterizan a la misma recta, se realiza un filtrado de estos segmentos, cuando sus extremos son muy similares o se realiza una fusión entre segmentos solapados parcialmente. Además se realiza un filtrado de segmentos por su longitud, despreciando aquellos inferiores a 50 píxeles.

2.1.3. Obtención de esquinas

Como se ha mencionado, cada segmento s está representado por su punto inicial (x_0, y_0) y su punto final (x_1, y_1) . A partir de ellos, se puede calcular la recta r a la que pertenecen:

$$s : (x_0, y_0)(x_1, y_1)$$

$$r : y = m \cdot x + b \text{ con } m = \frac{(y_1 - y_0)}{(x_1 - x_0)}$$

Si se agrupan los segmentos en horizontales y verticales se pueden combinar, siempre seleccionando uno de cada tipo, para encontrar los puntos de cortes entre ellos. Entre dos rectas coplanares existe un punto en el espacio donde se cortan (incluso las paralelas lo hacen en el infinito). Para calcular el punto de corte entre dos segmentos se calcula el punto de corte entre las rectas a las que pertenecen resolviendo el típico sistema:

$$r1 : y_c = m_1 \cdot x_c + b_1$$
$$r2 : y_c = m_2 \cdot x_c + b_2$$

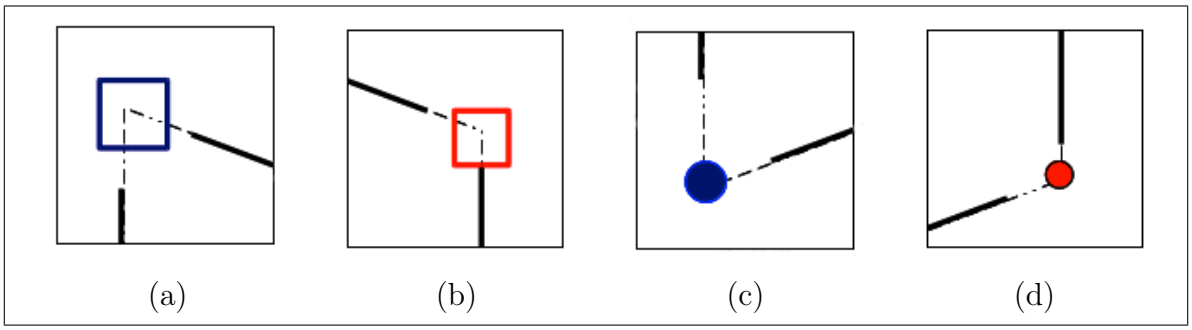


Figura 2.3: Tipos de esquinas detectadas: (a) Superior izquierda, (b) Superior derecha, (c) Inferior izquierda e (d) Inferior derecha

De todos los puntos de corte obtenidos sólo interesan aquellos que aparezcan dentro de la imagen y estén próximos a los extremos de los segmentos, ya que éstos son los que tienen más opciones de ser esquinas reales, y corresponder con las esquinas de los carteles. Dependiendo de la localización de éstas en relación a los segmentos se agrupan en cuatro tipos de esquinas, como puede verse en la Figura 2.3.

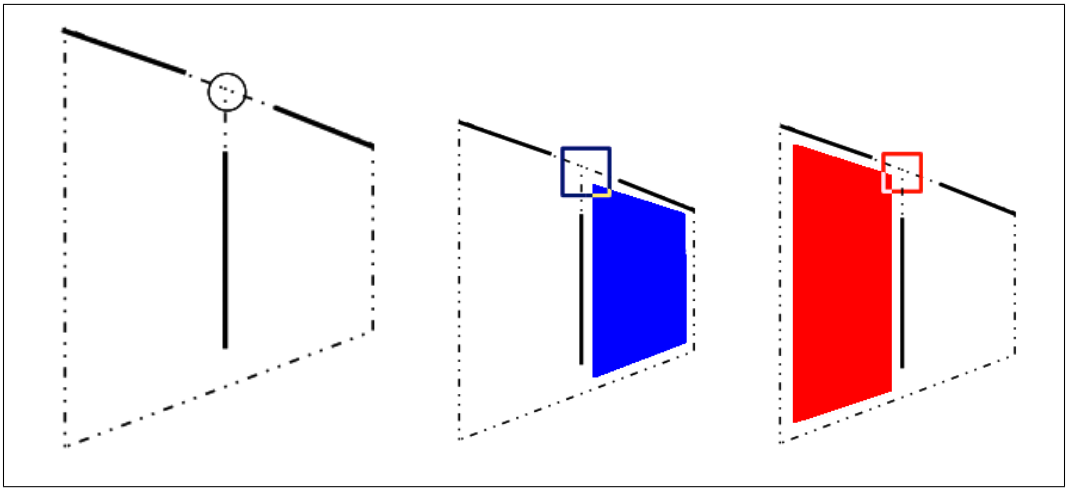


Figura 2.4: Situación en la que un punto de corte entre dos segmentos pertenece a varios tipos de esquinas, en este caso la esquina es tanto superior izquierda como superior derecha

En las situaciones en que el punto de corte cae en un zona más o menos central del segmento, si se considera que la esquina sólo puede pertenecer a un tipo, es posible perder algunas hipótesis interesantes. Por ello, cuando sucede esta situación se opta por añadir la esquina a todos los tipos que sea necesario. Un ejemplo de esta situación se ilustra en la Figura 2.4.

2.1.4. Búsqueda de hipótesis de rectángulos

Con las esquinas encontradas, se construyen las hipótesis de los rectángulos. Primero se intentan agrupar formando áreas rectangulares y finalmente, si las hipótesis han quedado incompletas, se estima como completarlas.

Alineación de esquinas. En primer lugar, para poder construir los rectángulos de la imagen es necesario buscar aquellas esquinas que sean candidatas a pertenecer a un mismo rectángulo y para ello se comprueba si están alineadas. Cada pareja alineada de esquinas debe cumplir las siguientes condiciones:

- Los segmentos alineados (horizontales o verticales, dependiendo del tipo de esquinas) deben pertenecer a la misma recta, como se ve en la Figura 2.5 (a).

- La posición relativa de las esquinas debe ser correcta, evidentemente, una esquina superior no puede pertenecer al mismo rectángulo que un esquina inferior que se encuentra por encima de ella. Un ejemplo de este caso puede verse en la Figura 2.5 (b).
- Debe existir una cierta distancia entre esquinas, ya que como hay esquinas incluidas en varios grupos, o incluso hay algunas que pueden estar demasiado cercanas, se comprueba que no se realicen alineamientos entre la misma esquina.

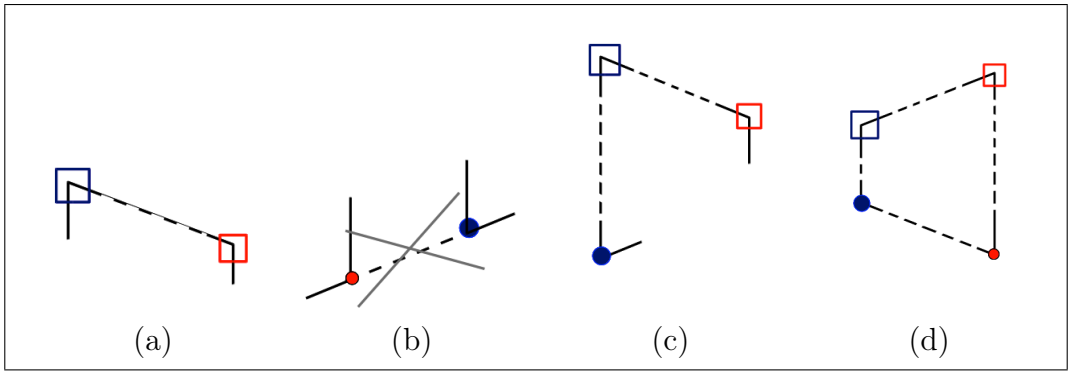


Figura 2.5: Agrupación de esquinas evaluando distintos criterios

Primero se buscan los alineamientos entre parejas, formando hipótesis de una alineación como en la Figura 2.5 (a). Sobre las hipótesis encontradas, se intentan forman ahora agrupamientos entre tres esquinas consiguiendo formar hipótesis de dos alineaciones, Figura 2.5 (c). Por último, se intentan alinear con la última esquina, formando hipótesis de cuatro alineamientos cuando se produce un doble alineamiento entre la nueva esquina y sus dos adyacentes, Figura 2.5 (d).

Las hipótesis se guardan con el mayor número de alineamientos obtenidos y en el caso de que alguna esquina no forme parte de ninguna hipótesis, se crea una nueva hipótesis que contiene únicamente a dicha esquina, formando las hipótesis más simples.

Completado de los rectángulos. Según el número de esquinas que se ha conseguido agrupar en cada hipótesis, existen rectángulos incompletos, exceptuando el caso de las hipótesis de cuatro alineamientos. Por lo tanto, se aproxima la localización del rectángulo completo. Dependiendo del número de esquinas agrupadas, se generan una o varias nuevas hipótesis. En la Figura 2.6 pueden verse ejemplos de cada tipo de hipótesis. En el caso de las hipótesis de tres alineamientos de esquinas se generan dos nuevas hipótesis de rectángulos como muestra la Figura 2.6 (a). En hipótesis de dos y uno se generan tres nuevos rectángulos Figuras 2.6 (c) y (d) respectivamente, aunque no siempre para esta última con el fin de evitar que aparezcan rectángulos muy desproporcionados. Y por último, en el caso de las hipótesis de las esquinas sin agrupar, se crea un único rectángulo Figura 2.6 (b).

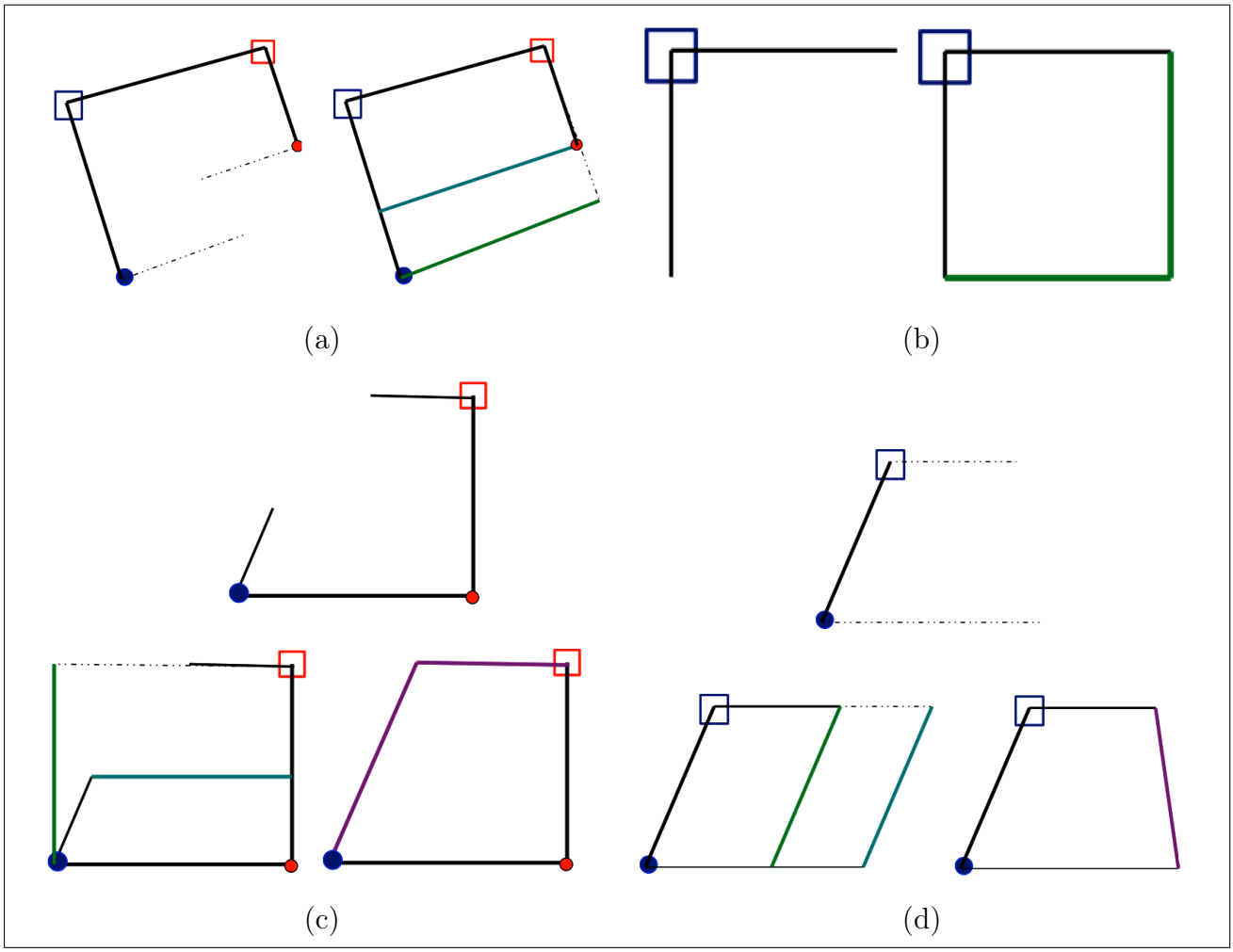


Figura 2.6: Hipótesis de rectángulos generados a partir de la agrupación de esquinas, en grupos de 1(d), 2(c) y 3(a) alineamientos o de esquinas sueltas (b)

2.2. Evaluación de las hipótesis

Junto con la posibilidad de que no todas las rectas y esquinas hayan podido ser detectadas correctamente, también es necesario barajar la posibilidad de que no todas las detectadas son de interés. Por tanto, es muy probable que las hipótesis generadas se hayan obtenido de rectas procedentes de otros elementos de la escena como puertas, ventanas, paredes de baldosas o ladrillo... en lugar de las que corresponden con los bordes del cartel, que son las que interesan. Por esto, es necesario desarrollar un proceso por el cual se pueda evaluar qué hipótesis de rectángulos son más probables de corresponder a un cartel y contener texto.

2.2.1. Filtrado de carteles

Para realizar esta evaluación, se han diseñado una serie de filtros para analizar el contenido dentro de cada hipótesis rectangular, para determinar si es probable que éste contenga texto o no. Para el análisis de su contenido se realizan observaciones sobre la región rectangular en escala de grises. El histograma de la imagen representa la frecuencia con la que los niveles de gris aparecen en la imagen. El eje de abscisas indica los distintos niveles (discretos) de grises y en el eje de ordenadas se representa la frecuencia, o el número de píxeles que poseen ese nivel de gris. El histograma proporciona información de cómo están distribuidos los distintos niveles de gris.

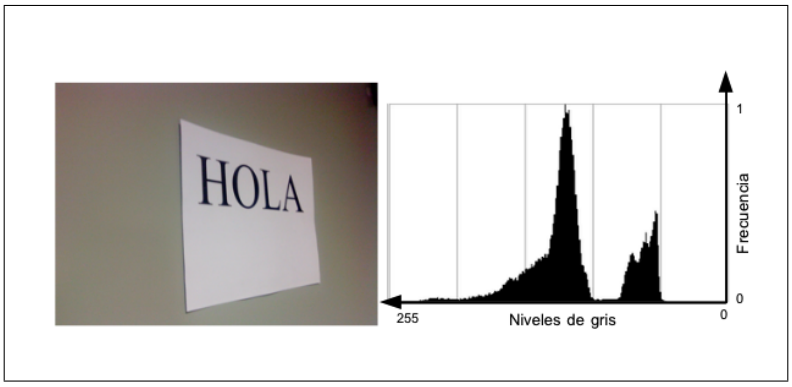


Figura 2.7: Imagen con su histograma

Para poder especificar los parámetros que debe cumplir un rectángulo que contenga texto, se han seleccionado unos cuantos ejemplos, positivos y negativos, y se han observado sus distribuciones. Se han observado las siguientes características:

- Un cartel típicamente tiene una distribución de colores para las letras muy contrastada con el color destinado para el fondo, para que sea lo más legible posible. Parece que la distribución de los píxeles de los rectángulos de carteles deberán tener claramente dos picos: uno correspondiente al texto y el otro al fondo.

- Un rectángulo con contenido uniforme, por ejemplo una baldosa, suele tener un único tono de gris dominante en el histograma, que corresponderá con la moda de la distribución. Como la mayoría de los píxeles se agrupan alrededor de la moda, el valor de la media y la moda serán muy próximos y además la desviación típica de la distribución suele ser pequeña. Otro indicador a evaluar, relacionado con este aspecto, es el coeficiente de Curtosis. Analiza el grado de concentración o dispersión que presenta una distribución. Si es leptocúrtica (el coeficiente es mayor que 0) los datos están muy concentrados o cercanos al promedio de la distribución.

Los carteles, normalmente, se espera que sean rectangulares, pero sin ser excesivamente alargado o aplastados. Por ello, se van a descartar aquellos rectángulos cuya relación entre anchura y altura no es adecuada, como por ejemplo los mostrados en la Figura 2.8.

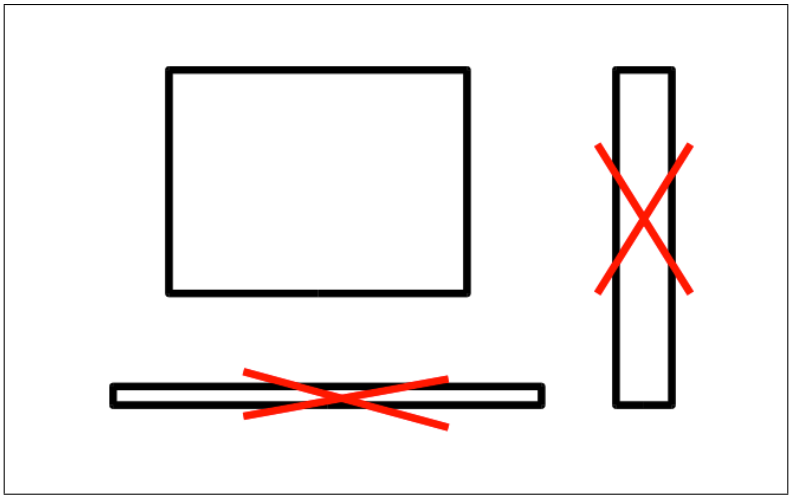


Figura 2.8: Rectángulo esperado y rectángulos descartados por su forma

Gracias a las anteriores observaciones sobre los valores de la distribución típica de grises en un cartel con texto, y sobre su forma, se diseñan los siguientes criterios. Para calcular la probabilidad de que un rectángulo tenga texto, se evaluan las siguientes expresiones:

$$(\bar{x} > 1) \wedge (M_o > 1) \tag{2.1}$$
$$(\sigma > 2) \tag{2.2}$$
$$(K < 1) \tag{2.3}$$
$$(Max_1(h) < 40\,\%) \tag{2.4}$$
$$(si\ \exists (Max_1(h) \wedge Max_2(h))) \tag{2.5}$$
$$(ratio \geq 0{,}5) \wedge (ratio \leq 6) \tag{2.6}$$

Donde $\bar{x}$, M_o y σ corresponden con la media, moda y desviación típica de la distribución respectivamente; h es el histograma de la distribución y $Max_i(h)$ corresponde máximo i-ésimo de los canales del histograma h y $ratio$ es la proporción entre los lados del rectángulo:

$$\bar{x} = \frac{\sum_i^n (x_i \cdot f_i)}{n} \tag{2.7}$$
$$\sigma = \sqrt{\frac{\sum_i^n (x_i - \bar{x})^2}{n-1}} \tag{2.8}$$
$$K = \frac{\mu_4}{\sigma^4} = \frac{\frac{1}{n} \cdot \sum_i^n (x_i - \bar{x})^4 \cdot n_i}{(\frac{1}{n} \cdot \sum_i^n (x_i - \bar{x})^2 \cdot n_i)^2} - 3 \tag{2.9}$$

$$ratio = \frac{ladoHorizontal}{ladoVertical} \tag{2.10}$$

Cada uno de los criterios anteriores que se cumple *vota* para determinar si la hipótesis analizada es texto o no. Más formalmente, y de cara a ordenar qué hipótesis son más probables, para seleccionar cuales van a continuar el proceso restante, se estima su probabilidad **P** como sigue:

$$\mathbf{P}(T \mid R_i) = \frac{F_i}{|F|} \tag{2.11}$$

La probabilidad de que la hipótesis rectangular R_i sea de la clase texto *(T)*, depende de cuantos *votos* recibe (F_i) del total de votos posibles $|F|$. $|F|$ es 13 ya que los criterios (2.6) y (2.5) tienen un peso de 7 y 2 respectivamente, mientras que el resto tienen peso 1. A los criterios (2.6) y (2.5) se les ha asignado un peso superior por considerarlos características de mayor importancia. Únicamente se aceptan aquellas hipótesis cuya probabilidad sea superior al 75 %.
En el caso de que existan demasiadas hipótesis caracterizadas como texto, sólo se considerarán aquellas de mayor **P**, con el fin de no sobrecargar el trabajo de la aplicación.

2.3. Proyección de hipótesis aceptadas

Llegado a este punto del proceso, se dispone del grupo de hipótesis de rectángulos encontrados en la imagen cuyo contenido es muy probable que sea texto. Como paso previo a reconocer los caracteres con un *OCR* se proyecta la imagen a una vista frontal con el fin de proporcionar al OCR una imagen más fácil de procesar.

Al fotografiar un mismo plano, desde distinto ángulo, se obtienen imágenes desde distintas perspectivas del mismo. En las distintas imágenes algunas propiedades geométricas se conservan, como la colinealidad (una línea recta se proyecta como una línea en las imágenes) mientras que otras no se mantienen, como el paralelismo. Las líneas paralelas de la realidad, dependiendo del ángulo al capturar la imagen, no se presentan como tal en la imagen. Por ello, es muy probable que los rectángulos obtenidos de la imagen seleccionada por el usuario correspondan más bien con cuadriláteros deformados; en el que sus lados enfrentados no sean paralelos ni los ángulos que forman sean de 90º, y por lo tanto, que el texto contenido en dichos rectángulos esté deformado también.

Una limitación impuesta por la mayoría de *OCRs* y en particular los de libre disposición, es que para llevar a cabo un correcto reconocimiento de los caracteres, el texto debe estar lo más frontal y paralelo a los bordes de la imagen posible. Por ello, se intenta transformar la imagen en otra (como se sugiere en la Figura 2.9) que muestre la escena

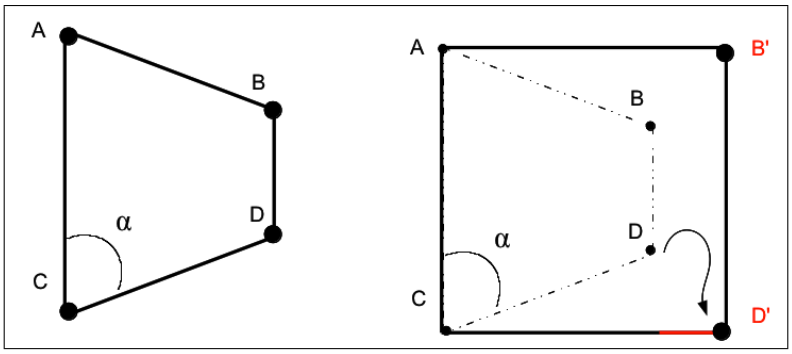


Figura 2.9: Vista actual y objetivo del rectángulo

como si ésta hubiera sido capturada de frente, para corregir la deformación presente en el texto en la medida de lo posible.

Como el área buscada corresponde al mismo plano, se puede realizar una transformación proyectiva entre la vista actual y la objetivo. En geometría proyectiva [17], esta transformación se representa mediante una homografía, que relaciona cada píxel del plano en una imagen con su proyección en la otra imagen. Para estimarla, basta con obtener puntos o rectas coplanares de la escena y sus correspondencias en la otra imagen. En nuestro caso, la segunda imagen es una vista "ideal" frontal que se quiere tener del cartel, así que se debe construir esta imagen "objetivo". En primer lugar, se estima que forma debería tener cada rectángulo posible si estuviera de frente. Por un lado se puede establecer que la altura vertical es la mayor de las alturas verticales observadas en la hipótesis $MAX(\bar{AC}, \bar{BD})$, y que la anchura será similar o algo mayor al lado horizontal $(\bar{CD})$ que se ve en la Figura 2.9. Según si la vista ya era casi frontal o no, este lado horizontal se debería ver de un tamaño u otro, pero siempre paralelo y de la misma longitud que el lado $(\bar{AB'})$. Experimentalmente se ha definido que la longitud de los lados estimados $(\bar{AB'})$ y $(\bar{CD'})$, según el ángulo α que forman con la vertical, se aproximen como sigue:

$$lado = \begin{cases} lado & \text{si} \quad \alpha > 83^o \\ lado \cdot 1{,}1 & \text{si} \quad 63^o < \alpha \leq 83^o \\ lado \cdot 1{,}2 & \text{si} \quad 46^o < \alpha \leq 63^o \\ lado \cdot 1{,}3 & \text{si} \quad 23^o < \alpha \leq 46^o \\ lado \cdot 1{,}5 & \text{si} \quad \alpha \leq 23^o \end{cases} \tag{2.12}$$

Con las coordenadas de las esquinas del rectángulo estimado, ya se dispone de las correspondencias entre las cuatro esquinas del rectángulo detectado (ABCD) y su estimación en posición frontal (AB'CD'). Estas cuatro correspondencias, son lo mínimo necesario para estimar la transformación (homografía) que se debe aplicar a todo el plano para pasar de una posición a otra. Un ejemplo de cómo una homografía relaciona dos planos puede

verse en la Figura 2.10.

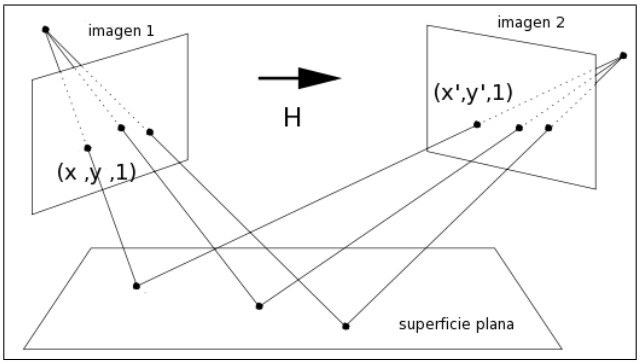


Figura 2.10: Ejemplo de una transformación proyectiva $x' = H \cdot x$

Dadas las correspondencias entre los puntos: $(A \to A, B \to B', C \to C, D \to D')$ cada punto **x'** se puede estimar a partir del punto original **x** como: **x'** $= H \cdot \mathbf{x}$ donde H es la matriz que define la homografía:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.13}$$

siendo h_{ij} los elementos de la matriz H, y representando los puntos con coordenadas homogéneas $[x, y, 1]$ [17]. Cada punto (x', y') del plano se calcula como:

$$x' = \frac{h_{11} \cdot x + h_{12} \cdot y + h_{13}}{h_{31} \cdot x + h_{32} \cdot y + h_{33}}, \quad y' = \frac{h_{21} \cdot x + h_{22} \cdot y + h_{23}}{h_{31} \cdot x + h_{32} \cdot y + h_{33}} \tag{2.14}$$

Estas ecuaciones son lineales en los elementos de H. Con cuatro correspondencias de puntos se generan ocho ecuaciones lineales, por lo que son suficientes para resolver H multiplicada por un factor insignificante. La única restricción es que no puede haber tres puntos colineales, es decir, tres puntos pertenecientes a la misma recta.

La relación entre imágenes establecida por una homografía es válida si la escena es plana, situación correcta en este caso ya que se está proyectando un cartel. En la Figura 2.11 pueden observarse ejemplos de las proyecciones conseguidas, proyectando cada píxel con la H estimada e interpolando aquellas posiciones desconocidas.

2.4. Filtrado de hipótesis proyectadas

Como paso previo al reconocimiento final de los caracteres, todos los *OCRs* binarizan la imagen con el fin de separar los caracteres del fondo. Tras realizar varias pruebas, se

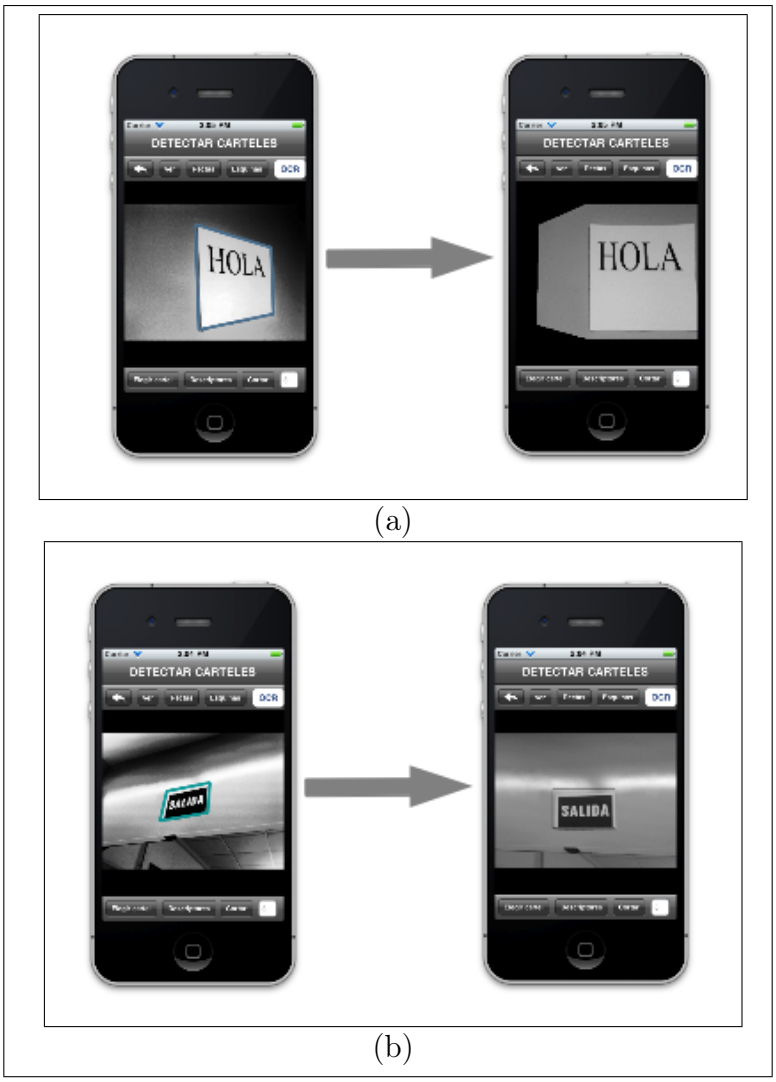


Figura 2.11: Ejemplos de la proyección realizada con las hipótesis aceptadas

ha observado que esta binarización es estática y no siempre produce buenos resultados, es por ello, que como paso final antes del *OCR* se va a realizar una binarización adaptativa de la imagen para facilitar la tarea al *OCR* y que éste obtenga mejores resultados.

2.4.1. Binarización de la imagen

Como se supone que el color del texto del cartel está muy diferenciado con el del fondo, en este paso se va a binarizar la imagen para intentar separar el fondo de las letras. La binarización se basa en una umbralización del histograma.
Para obtener unos resultados satisfactorios, dependiendo de la imagen, se ha seleccionado un umbral distinto según el valor de la moda del histograma. Ya que si se fija un umbral

fijo para todas las imágenes, es posible que en algunas se elimine el texto o que en la imagen resultante no se distingan los caracteres a causa del ruido. Dado que en los carteles con texto, al fondo le corresponden más píxeles, se estima que el nivel de gris de la moda corresponde con el del fondo.

La binarización se realiza de la siguiente manera:

$$I(i,j) = \begin{cases} 1\,(BLANCO) & \text{si} \quad I(i,j) \geq U \\ 0\,(NEGRO) & \text{en caso contrario} \end{cases} \tag{2.15}$$

donde $I(i,j)$ corresponde al nivel de gris del píxel (i,j), y donde el umbral U depende de la moda M_o:

$$U = \begin{cases} 150 & \text{si } M_o < 7 \\ 85 & \text{si } M_o \geq 7 \end{cases} \tag{2.16}$$

donde los niveles de gris han sido discretizados en 18 valores.

2.4.2. Análisis de trazos rectos en hipótesis proyectadas

Para realizar este análisis se emplea el mismo extractor implementado en la sección 2.1.2 pero extrayendo rectas de hasta 5 píxeles para evaluar los siguientes puntos:

Descarte de carteles. En el caso de no haberse detectado como mínimo 10 rectas, es muy probable que no aparezca texto en la imagen. Este umbral se ha establecido experimentalmente tras observar el número de rectas obtenidas en distintos ejemplos. Podría parecer que este umbral pueda descartar carteles cuya tipografía es muy redondeada pero se ha observado que no suele pasar, ya que siguen apareciendo segmentos de rectas en los bordes de las letras debido a que el límite de longitud es muy bajo. En la Figura 2.12 (b) se ven los segmentos horizontales en verde y los verticales en rojo que aparecen en un cartel con texto.

Eliminación posible borde. Es posible que como consecuencia, de la proyección o el ruido, aparezca un borde alrededor de la imagen final. Para evitar que este borde provoque errores en el reconocimiento de caracteres, se trata de eliminar. Para detectar si existen bordes sobre la imagen, se buscan rectas horizontales y verticales, de tamaño similar al de la imagen, próximas a los límites de la imagen para recortar por ellas, como se ve en la Figura 2.12 (a) las líneas en azul.

2.5. Integración del proceso desarrollado y un OCR sobre el *iOS*

Para completar la aplicación, se ha realizado la integración de un *OCR* de código disponible, buscando código libre en un lenguaje apropiado para integrarlo al proyecto

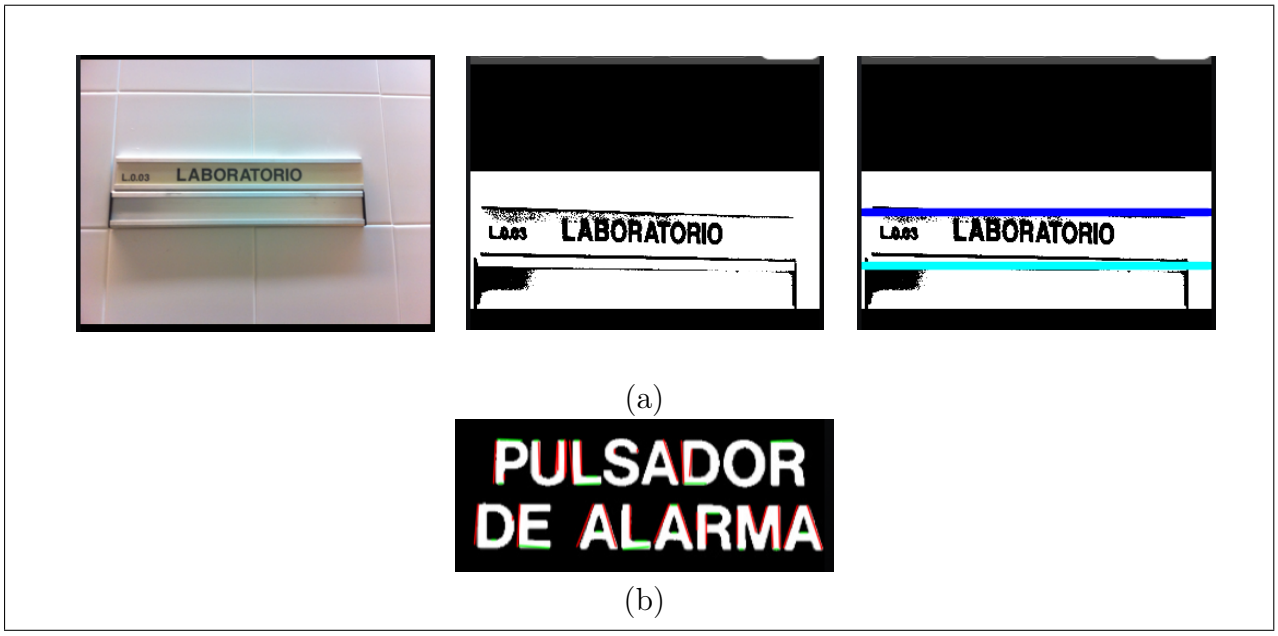


Figura 2.12: Extracción de rectas para eliminar el borde existente en la imagen (a) y comprobar la existencia de letras dentro de un cartel (b)

implementado sobre *iOS* del *iPhone*.

Se han evaluado los siguientes *OCRs*:

- OCRAD[1]: implementado en C++
- TESSERACT[2]: implementado en C++
- GOCR[3]: implementado en C

Tras realizar varios tests con imágenes de prueba, el OCRAD y el TESSERACT tienen un comportamiento similar, mejorando al GOCR. De los dos anteriores, se ha seleccionado el TESSETACT por ofrecer más fácil integración con nuestro proyecto, además de permitir cambiar el idioma del texto a detectar. En el Anexo D puede verse los detalles de la comparativa realizada.

[1]http://www.gnu.org/software/ocrad/
[2]http://code.google.com/p/tesseract-ocr/
[3]http://linux.die.net/man/1/gocr

Capítulo 3

Diseño de la aplicación desarrollada y validación de la misma

Una vez diseñado el proceso necesario, para transformar la imagen en otra que permita a un OCR extraer su texto más fácilmente, se aborda el desarrollo del prototipo final. El funcionamiento de la aplicación puede separarse en tres pasos, tal y como muestra la Figura 3.1:

1. Elección de una imagen
2. Procesamiento de la imagen
3. Lectura y muestra del texto

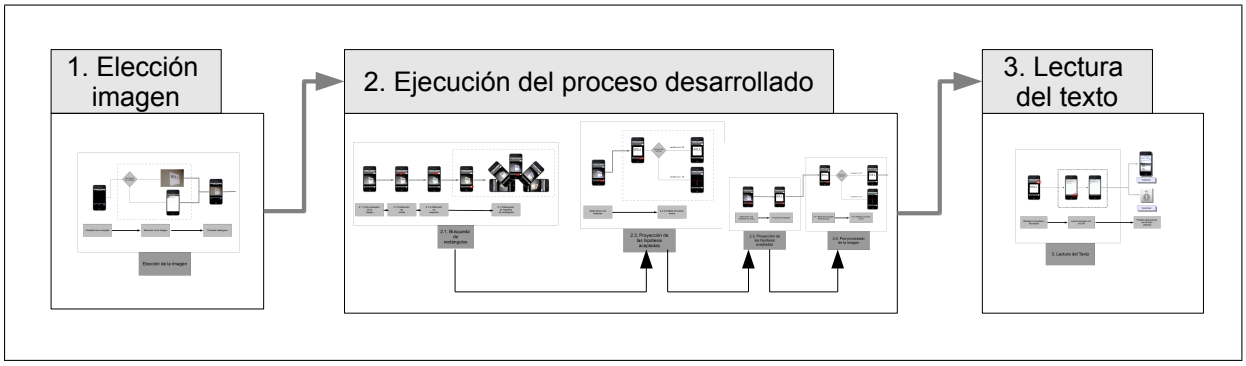


Figura 3.1: Funcionamiento básico de la aplicación. Tras seleccionar una imagen, la aplicación será capaz de extraer su texto

En el primer paso, y para poder continuar, el usuario debe seleccionar la imagen a procesar. Dicha imagen puede elegirse de entre las ya existentes en la biblioteca del dispositivo o por el contrario, puede capturarla al instante gracias a su cámara. En caso de que el dispositivo no tuviera cámara, como por ejemplo algunos modelos del *iPod Touch*,

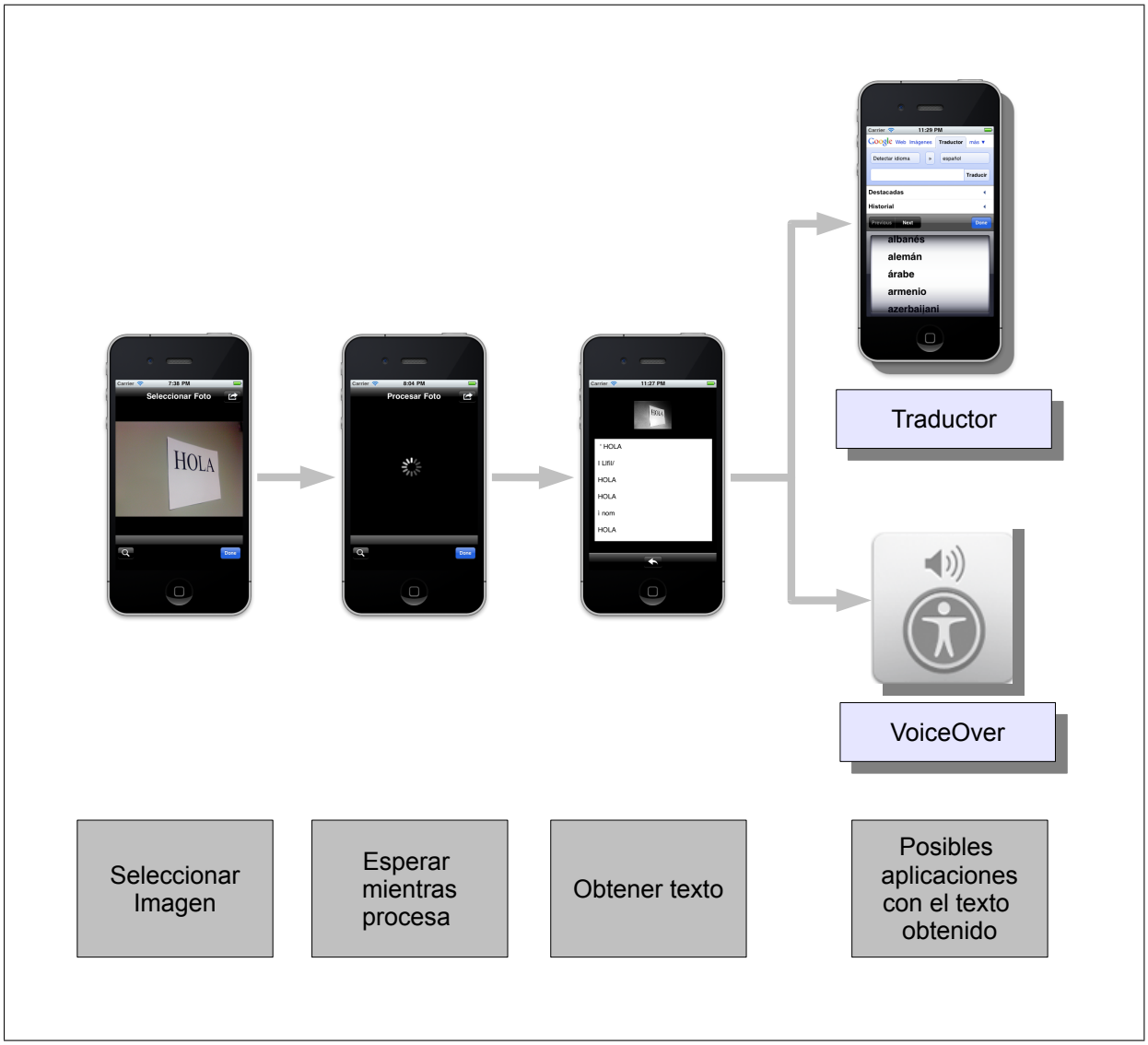


Figura 3.2: Pantallas de la aplicación final

esta posibilidad no se permite. Tras la elección de la imagen se lleva a cabo todo el proceso descrito en el Capítulo 2. La realización de este proceso es transparente al usuario, el cuál debe permanecer a la espera de que concluya. Para acabar, se muestran los resultados obtenidos en la pantalla. Una vez obtenido el texto de la imagen puede utilizarse para otro tipo de aplicaciones: permitir hacer una consulta a un traductor online, como *Google Translator* o podría ser reproducido para escuchar el texto sin necesidad de verlo. La interfaz de la aplicación final puede verse en la Figura 3.2.
Como un posible uso de esta aplicación es prestar asistencia a personas con algún tipo de discapacidad visual, en un primer momento se pensó incluir un sintetizador de voz que

fuera capaz de leer los resultados mostrados. Pero dadas las características de accesibilidad disponibles en el *iPhone 3GS*, *iPhone 4* y en el *iPod Touch* (3ª generación y posteriores) no es necesario. *VoiceOver*[1] es un lector de pantalla instalado por defecto que se controla únicamente con gestos sobre la pantalla táctil, por lo cual, sólo ha sido necesario adaptar el diseño para que fuera compatible con este sistema.

3.1. Verificación del funcionamiento de la aplicación

Para garantizar que la aplicación desarrollada es funcional se han realizado múltiples pruebas con imágenes reales, tomadas con el móvil, para comprobar su correcto funcionamiento, tanto en el simulador como en dos dispositivos físicos. Para realizar una evaluación del prototipo, se ha realizado una comparación entre los resultados que aporta un *OCR* por sí solo y los que aporta ese mismo *OCR* una vez integrado en el proceso realizado en este proyecto.

3.1.1. Pruebas unitarias

A la hora de desarrollar una aplicación para un dispositivo físico hay que tener especial cuidado con los recursos de los que se disponen, es por ello, que cada paso implementado del proceso ha sido comprobado individualmente en el simulador del *iPhone* para garantizar que se ejecutaba en un tiempo razonable. Además, se ha realizado pruebas con varias imágenes para verificar que el comportamiento era el esperado. Para facilitar estas pruebas unitarias se ha desarrollado una interfaz secundaria, con la que ha sido más fácil localizar y subsanar de errores existentes en las fases iniciales del proceso y así evitar su acumulación. En el Anexo E puede verse con más detalle el diseño de esta interfaz.

Estas pruebas también se han realizado sobre dispositivos reales con el fin de evitar futuras complicaciones o posibles errores de compatibilidad entre el simulador y el dispositivo, o posibles ejecuciones excesivamente lentas.

3.1.2. Pruebas de validación

Al describir los objetivos de este proyecto se comparaba el funcionamiento de un *OCR* ante dos imágenes: una con el texto frontal y perfectamente encuadrado y otra en la que se observaba un cartel desde mayor distancia y con una cierta perspectiva, Figura 1.2. Con esta última imagen el *OCR* era incapaz de extraer su texto. Si ahora, una vez finalizado este proyecto, se ejecuta la aplicación desarrollada con esa imagen puede verse en la Figura 3.3 (a) que esta vez sí se ha conseguido extraer su texto (b). Tal y como se refleja en la misma figura, también es posible detectar el texto cuando existe más de un cartel como en (c).

[1] http://www.apple.com/es/accessibility/voiceover/

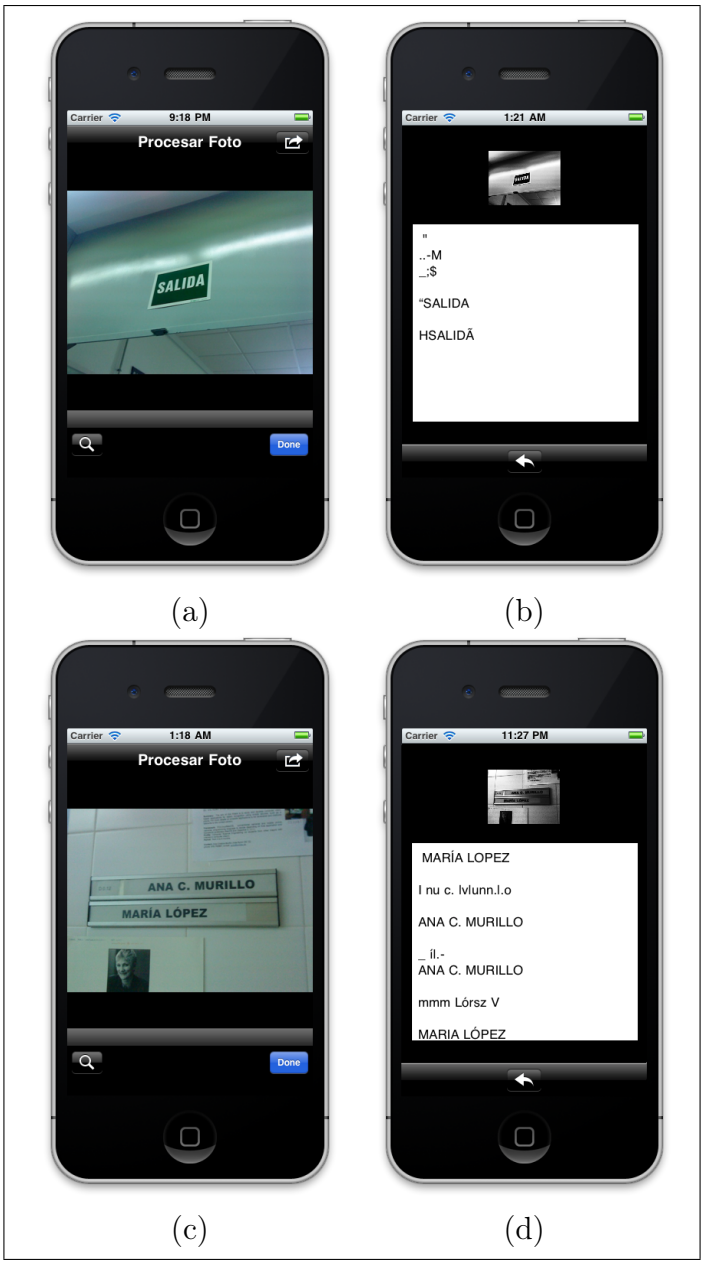


Figura 3.3: Demostración aplicación. A partir de la imagen (a) se obtiene el texto mostrado en (b). Incluso, cuando hay 2 carteles como en (c), se detectan el texto de los 2 (d)

3.2. Resultados del proyecto

Para medir de manera más objetiva las mejoras y rendimiento de la aplicación diseñada, se ha realizado la misma comparación de la Figura 3.3 con 30 imágenes, 16 imágenes frontales y 14 no frontales. Se comparan los resultados obtenidos con la aplicación desarro-

	vistas frontales		vistas NO frontales	
	PFC	OCR	PFC	OCR
	μ | σ	μ | σ	μ | σ	μ | σ
% lineas	91,67 % | 0,260 %	57,50 % | 0,500 %	60,71 % | 0,490 %	28,57 % | 0,47 %
% char	76,83 % | 0,290 %	39,71 % | 0,400 %	39,35 % | 0,400 %	9,030 % | 0,18 %
% éxitos	68,75 % | − − −	25,00 % | − − −	30,76 % | − − −	0,000 % | − − −
% fracasos	6,250 % | − − −	37,50 % | − − −	46,15 % | − − −	76,92 % | − − −

Cuadro 3.1: Tabla que contiene un resumen de los resultados obtenidos por el OCR con y sin añadir el proceso propuesto en este proyecto

llada y con los obtenidos con el mismo *OCR* pero sin el proceso propuesto en este proyecto.

Para cada imagen se va a contabilizar el número de líneas y caracteres que se han acertado para calcular una tasa de aciertos. Se han tenido en cuenta las siguientes consideraciones:

1. Cuando en una imagen, no se consigue reconocer ni una sílaba del texto buscado, se considera que no se ha reconocido ningún carácter y que todo es ruido.
2. Para que una línea se considere acertada debe existir al menos una sílaba de la línea correspondiente de la imagen.
3. Si se han detectado más líneas de texto, además de las correctas, se consideran falsos positivos, es decir, ruido.
4. Un carácter se ha acertado si es el mismo que el del texto de la imagen, es decir, si entre el carácter reconocido y el del texto lo único que varía es su acento o si es mayúscula o minúscula. Se considera acierto ya que su lectura no conduce a error.
5. Debido a los problemas en la distinción entre la "o" y el "0", si se detecta alguno de los caracteres anteriores en respuesta al otro, se considera acierto.
6. Una imagen se considera que ha sido reconocida con éxito si se reconocen correctamente su número de líneas y más del 70 % de sus caracteres. Por el contrario, sólo será considerada como fracaso si no se ha conseguido reconocer ni una línea, en otras palabras, no se ha conseguido reconocer ni una sílaba del texto.

Las tablas con los resultados individuales de cada imagen pueden consultarse en el Anexo F, donde también pueden verse el conjunto de imágenes de prueba. Para realizar

dichas pruebas, las imágenes se han dividido en dos grupos pensando en la facilidad que tendrá el OCR al reconocer su texto. El primer grupo contiene las imágenes cuyo cartel o carteles se encuentran de frente, y en el otro, aquellas en las que éstos se encuentran inclinados y/o deformados. En la tabla 3.2, la columna *OCR* contiene los resultados proporcionados por el *OCRbase*, mientras que la columna *PFC* muestra los datos obtenidos por el mismo OCR pero integrado junto con el proceso desarrollado en este proyecto. Esta tabla recoge la media y la desviación típica de los resultados obtenidos con las imágenes de prueba siguiendo los criterios explicados anteriormente. Para realizar una comparación se calculan los siguientes porcentajes:

- Líneas acertadas
- Caracteres acertados
- Imágenes reconocidas con éxito
- Fracaso en el reconocimiento

Observando los datos se comprueba que, como era de esperar, el *OCRbase* por si solo, consigue mejores resultados cuando las imágenes están de frente. También se observa que todos los porcentajes de aciertos del *OCRbase* son inferiores a cuando éste esta integrado en el proceso desarrollado, es decir, que con este proyecto se ha conseguido reconocer más caracteres que antes. Pero lo interesante de estos resultados no es sólo su aumento, sino que se ha conseguido, en un 50 %, un éxito en imágenes que con el *OCRbase* eran fracasos. Esto puede verse en los resultados del Anexo F. Existe un único caso en el que el proceso fracasa mientras que el *OCRbase* es capaz de reconocer algo, sin llegar a ser un éxito. Esto se produce por una mala binarizacion durante el proceso como se refleja en la Figura 3.4.

En resumen, con este proceso se ha conseguido casi un 70 % de éxitos frente al 25 % que proporciona el *OCRbase*. El porcentaje de no éxitos de este proyecto aún se podría reducir, ya que existen casos en los que se realiza una mala binarización, como ya se ha visto en el ejemplo anterior de la Figura 3.4, y otros en los que el *OCR* no es capaz de reconocer correctamente el texto aún estando la imagen perfectamente segmentada y binarizada como se observa en la Figura 3.5. Para concluir, se puede afirmar que el proceso implementado en este proyecto cumple su propósito general: extraer el texto de imágenes que un OCR común no consigue obtener.

Cabe destacar que los resultados se han obtenido con el simulador, siendo posible que en función de la versión y/o resolución del dispositivo físico utilizado, las imágenes seleccionadas pueden ser redimensionadas automáticamente, y por lo tanto ofrecer algún tipo de variación en los resultados.

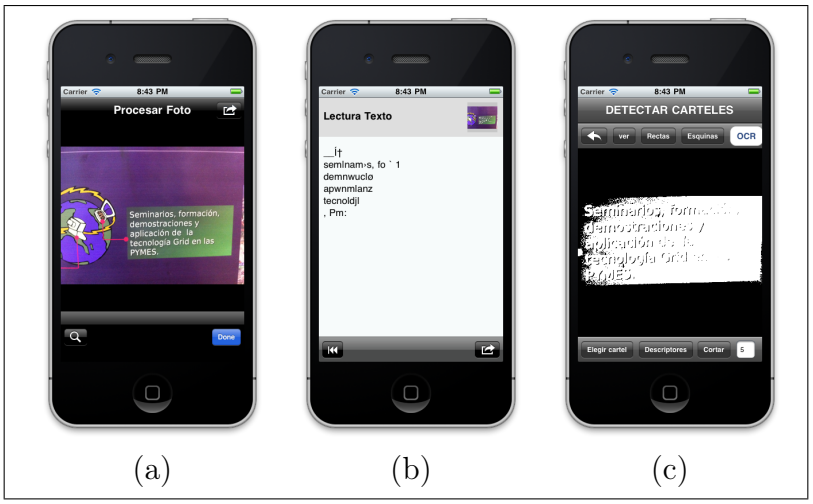


Figura 3.4: (a) Imagen de prueba en la que el OCR consigue reconocer más caracteres (b) frente al fracaso del PFC por una mala binarización (c)

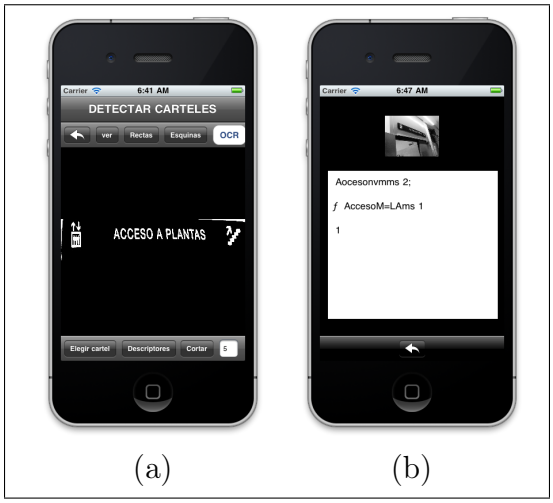


Figura 3.5: (a) Aunque el cartel se ha detectado y binarizado correctamente, el OCR no es capaz de reconocer su texto (b)

Capítulo 4

Conclusiones y valoración personal

Este capítulo contiene las conclusiones extraídas tras la realización de este Proyecto Final de Carrera y una propuesta de posibles líneas de trabajo futuro. Al final se presenta una valoración de lo que su realización ha supuesto a nivel personal.

4.1. Conclusiones

El avance en la tecnología utilizada en teléfonos móviles abre todo un mundo nuevo de posibilidades a la hora de desarrollar aplicaciones para estos dispositivos. La calidad que ofrecen sus cámaras, así como el incremento en la capacidad de cómputo, permiten construir aplicaciones que hace unos años eran impensables debido a su consumo de recursos. Gracias a ello y con el deseo de crear una aplicación para un teléfono móvil, combinada con técnicas del campo de la visión por computador, se ha desarrollado este proyecto.

Al comienzo de este proyecto se esperaba construir un aplicación capaz de extraer el texto de una imagen mejorando los resultados que presentan los actuales reconocedores de caracteres. Para ello, se ha desarrollado todo un proceso a través del cual se obtiene el texto de un cartel, o carteles, con forma rectangular, que puede encontrarse en cualquier parte de una imagen. El hecho de liberar al usuario de la obligación de que el texto a leer debe estar cercano, de frente y encuadrado, permite el diseño de una aplicación mucho más robusta que incluso podría ayudar a personas con algún tipo de deficiencia visual.

Con el proceso desarrollado se ha conseguido proporcionar al *OCR* únicamente aquellas zonas que poseen una mayor probabilidad de ser un cartel y contener texto, para lo que se ha segmentando la imagen en rectángulos. Con este objetivo se aproximan los contornos de los objetos de la imagen en rectas. En un primer momento, para la extracción de rectas, se utilizó una función proporcionada por *OpenCV*, pero tras observar que los resultados no eran satisfactorios ni tampoco las soluciones diseñadas para subsanarlos, se optó por implementar otro extractor. Usando las rectas obtenidas por el extractor se construyó un detector de hipótesis de rectángulos. A su vez se ha diseñado un modelo que evalúa eficazmente aquellos rectángulos cuyo contenido parezca o no texto. Tras el

estudio, investigación y evaluación de diversas técnicas, finalmente se ha optado por basar el proceso en la forma y la distribución de los niveles de gris en la hipótesis.

Como es posible que la imagen hayan sufrido deformaciones a causa de la perspectiva y además una limitación de los *OCRs* es que para un correcto reconocimiento de caracteres éstos deben estar de frente, se ha diseñado como proyectar los rectángulos a través de una homografía. Con el mismo objetivo de mejorar los resultados obtenidos por el *OCR* se ha hecho una binarización adaptativa de la imagen, ya que aunque el *OCR* ya realiza una, ésta es fija y provoca pérdidas o introduce demasiado ruido en los caracteres en algunas imágenes. Como paso final, se ha realizado un análisis de trazos rectos de las imágenes binarizadas, para comprobar que realmente tienen texto y no ralentizar el proceso de lectura de rectángulos sin caracteres.

Con el proceso desarrollado se ha conseguido implementar una aplicación real y funcional que además cumple con los objetivos marcados en el proyecto: consigue extraer texto de carteles en imágenes que un *OCR* base no consigue hacerlo. Con objeto de validar el proceso, se ha llevado a cabo una evaluación exhaustiva con imágenes reales.

4.2. Trabajo futuro

Tras la realización del presente proyecto, se proponen a continuación algunas posibles líneas de trabajo futuro:

- Mejorar el modelo desarrollado de evaluación de hipótesis. Éste sólo analiza los niveles de gris, la forma y la existencia de trazos rectos. Podría completarse con más medidas como la distribución y orientación de los trazos verticales y horizontales, y el estudio de la distribución del gradiente y su orientación en imágenes de texto. También podría estudiarse la viabilidad de integrar un método para que la aplicación fuera aprendiendo a reconocer los carteles conforme su uso, con técnicas de aprendizaje automático.
- Mejorar el *OCR* integrado para conseguir un mejor reconocimiento o realizar consultas de diccionario con las cadenas obtenidas con el fin de completar o rectificar cadenas incompletas o mal detectadas.
- Desarrollar una aplicación todavía más funcional. El proceso podría realizarse sobre una captura de vídeo en tiempo real, de esta manera se podría avisar al usuario cuando se detectara un posible texto.

4.3. Valoración personal

La realización de este proyecto ha ampliado mis conocimientos, desde darme la oportunidad de aprender a manejar un nuevo sistema operativo, entorno y lenguaje de pro-

gramación hasta conceptos mucho más teóricos sobre la visión por computador. También, explorar el campo de la programación sobre dispositivos móviles me ha aportado una visión mucho más cuidadosa hacia la programación, ya que en estos dispositivos, los recursos son limitados.

A nivel personal he aprendido a ser constante y a dar la importancia que requiere al diseño de las aplicaciones. He perdido el miedo a enfrentarme con lo desconocido y a ser más paciente a la hora de obtener unos buenos resultados. Pero sobretodo, que el esfuerzo invertido ha merecido la pena.

Glosario de acrónimos

PDA Personal Digital Assistant Ordenador pequeño, ligero y portátil diseñado para realizar las tareas de una agenda electrónica (calendario, lista de contactos, bloc de notas y recordatorios) y con un sistema de reconocimiento de escritura.

GPS Global Positioning System Sistema global de navegación por satélite que permite determinar en todo el mundo la posición de un objeto, una persona o un vehículo con una precisión hasta de centímetros.

qwerty Tipo de teclado. Hace referencia a las primeras seis letras que aparecen en la esquina superior izquierda de este tipo de teclados.

Wi-Fi Wireless Fidelity Conjunto de estándares para redes inalámbricas basado en las especificaciones IEEE 802.11 (especialmente la 802.11b), creado para redes locales inalámbricas, pero que también se utiliza para acceso a internet.

OCR Optical Character Recognition Software encargado de reconocimiento óptico de caracteres, extrayendo de una imagen los caracteres de un texto y los guarda en un formato adecuado para su manipulación o consulta.

SDK Software Development Kit Conjunto de herramientas de desarrollo software permite a un programador crear aplicaciones para un sistema concreto. Las herramientas más comunes incluyen soporte para la detección de errores de programación así como un Integrated Development Environment (IDE). Los SDK frecuentemente incluyen, también, códigos de ejemplo y otra documentación de soporte.

IDE Integrated Development Environment Entorno de desarrollo integrado que permite escribir, compilar, ejecutar y depurar el código desarrollado.

CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart Prueba de Turing pública y automática para diferenciar máquinas y humanos.

Bibliografía

[1] J. Herrero. Reconocimiento visual con plataforma iphone: un traductor de japonés, Febrero 2010. Proyecto Fin de Carrera.

[2] S. Mori, C.Y. Suen, and K. Yamamoto. Historical review of ocr research and development. *Proceedings of the IEEE*, 80(7):1029 –1058, jul 1992.

[3] Victor Wu, R. Manmatha, and Edward M. Riseman. Finding text in images, 1997.

[4] M. Cord J. Fabrizio and B. Marcotegui. Text extraction from street level images. *City Models, Roads and Traffic (CMRT)*, 2009.

[5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. 10.1007/BF00994018.

[6] A.K. Jain and Bin Yu. Automatic text location in images and video frames. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1497 –1499 vol.2, aug 1998.

[7] Paul Clark and Majid Mirmehdi. Combining statistical measures to find image text regions. In A Sanfeliu, J J Villanueva, M Vanrell, R Alquezar, J-O Eklundh, and Y Aloimonos, editors, *Proceedings of the 15th International Conference on Pattern Recognition*, pages 450–453. IEEE Computer Society, September 2000.

[8] Huiping Li, D. Doermann, and O. Kia. Automatic text detection and tracking in digital video. *Image Processing, IEEE Transactions on*, 9(1):147 –156, jan 2000.

[9] Qixiang Ye, Jianbin Jiao, Jun Huang, and Hua Yu. Text detection and restoration in natural scene images. *Journal of Visual Communication and Image Representation*, 18(6):504 – 513, 2007.

[10] Xiangrong Chen and A.L. Yuille. Detecting and reading text in natural scenes. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–366 – II–373 Vol.2, june-2 july 2004.

[11] Q. Li C. Yang J. Liu N. Xu M. Yi K. Cai B. B Zhu, J. Yan. Attacks and design of image recognition captchas. *Proceedings of the 17th ACM conference on Computer and communications security*, 2010.

[12] A. C. Murillo, J. Košecká, J. J. Guerrero, and C. Sagüés. Visual door detection integrating appearance and shape cues. *Robotics and Autonomous Systems*, 56(6):pp. 512–521, June 2008.

[13] J. González Jiménez. *Visión por computador*. Paraninfo, ISBN: 84-283-2630-4, 1999.

[14] R. Kasturi R. Jain and B.G. Schunck. *Machine Vision*. McGraw-Hill, ISBN: 0-07-032018-7, 1995.

[15] J.B. Burns, A.R. Hanson, and E.M. Riseman. Extracting straight lines. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(4):425–455, 1986.

[16] J. Košecká and W. Zhang. Video compass. *ECCV '02 Proceedings of the 7th European Conference on Computer Vision*, pages 476–490, 2002.

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[18] D. Marr and E. Hildret. Theory of edge detection. *Proceedings of the Royal Society of London*, 207(1167):187–217, 1980.

[19] J. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

Apéndices

Apéndice A
Herramientas utilizadas

Uno de los objetivos de este proyecto ha consistido en la familiarización con el entorno de programación: *Mac OS, iOs, X-Code* y el simulador del *iPhone*, así como el aprendizaje del lenguaje de programación *Objective-C*. También se ha estudio la librería de código abierto *OpenCV* para facilitar las tareas relacionadas con la visión por computador. Con más detalle, a continuación se nombran algunas de las herramientas empleadas para el desarrollo de este proyecto.

Para poder construir aplicaciones para el *iPhone*, la empresa *Apple* proporciona un Software Development Kit (SDK) que permite desarrollar, probar, depurar y simular las aplicaciones. Las aplicaciones desarrolladas funcionan tanto en el *iPhone* como en el *iPad* y en el *iPod Touch* (productos también de la misma empresa). Dentro del SDK se dispone de tres herramientas:

- Xcode: es un IDE. Para desarrollar aplicaciones para el *iPhone*, el lenguaje de programación empleado debe ser Objective-C. Este entorno únicamente funciona en sistemas operativos MAC.
- Interfaz Builder: es una herramienta para el desarrollo visual de la interfaz gráfica de las aplicaciones.
- Instruments: es una potente herramienta de depuración y análisis del rendimiento de las aplicaciones que se están ejecutando, ya sean desde el simulador o desde el propio teléfono. Permite averiguar el consumo de memoria, de batería, el uso de la red...

Una vez desarrollada y probada la aplicación en el simulador, el siguiente paso es instalarla en el dispositivo físico. Una restricción que impone *Apple* es que no permite instalar aplicaciones que no están firmadas por ellos, de modo que, es necesario inscribirse en el programa de desarrolladores de *Apple* para obtener un certificado que te proporcione los permisos necesarios. Como paso final en el desarrollo de este tipo de aplicaciones, *Apple* pone a disposición de los desarrolladores una tienda online llamada App Store, donde se pueden subir estas aplicaciones para su venta. Sin embargo, por las restricciones de la

Figura A.1: Herramientas principales utilizadas

licencia para educación, el proceso de desarrollo termina con la instalación en un dispositivo físico. En particular se han realizado pruebas en un *iPhone* 4 y en un *iPod Touch*.
Debido a que en este proyecto se centra en temas de visión por computador, se hace uso de la biblioteca externa, *OpenCV*, con el propósito de facilitar las tareas relacionadas con el manejo de imágenes. *OpenCV* es una biblioteca de código libre[1]dedicada a la visión por computador. Está desarrollada en C y C++ además de ser multiplataforma, ya que existen versiones para *GNU/Linux*, *Mac OS X* y *Windows*. En este proyecto se usa su versión desarrollada en C para *Mac OS*.
Un esquema de las herramientas empleadas para el desarrollo de este proyecto pueden verse en la Figura A.1.

[1]Disponible en http://opencv.willowgarage.com/wiki/

Apéndice B

Gestión del tiempo

A continuación, en la Figura B.1 se muestra las tareas más importantes realizadas para llevar a cabo este proyecto y en la Figura B.2 el tiempo dedicada a cada una de ellas.

Figura B.1: Tareas desarrolladas en este proyecto

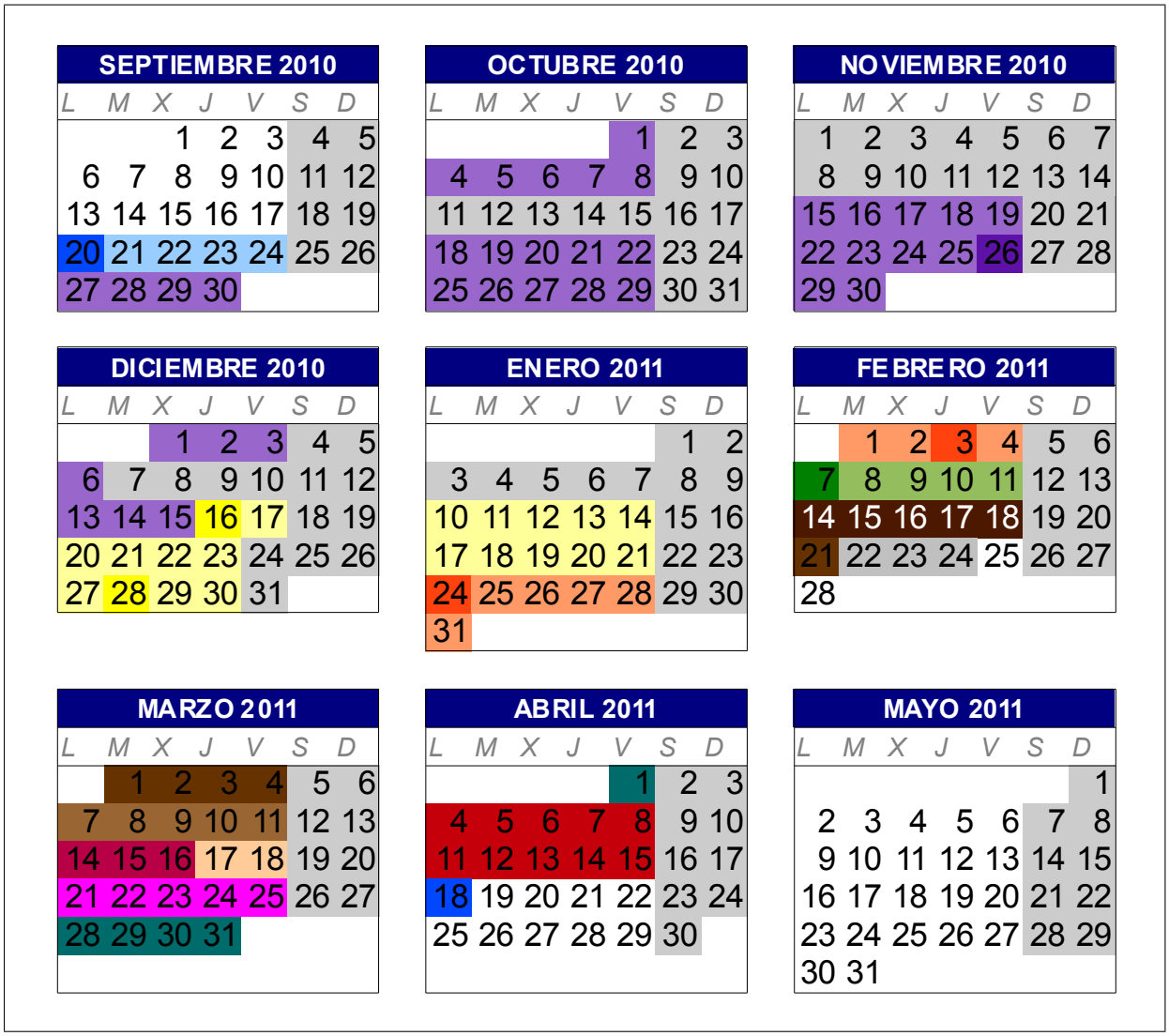


Figura B.2: Tiempo dedicado a las tareas desarrolladas en este proyecto

Apéndice C

Obtención de contornos

Un contorno o borde es una región de la imagen donde existe un cambio fuerte en el nivel de gris entre los píxeles adyacentes, es decir, donde existen un cambio de intensidad abrupta. Su causa principal es la intersección entre objetos, que al tener distintos niveles de reflectancias y al ser éstos proyectados sobre la cámara, generan discontinuidades de intensidad. Aunque también aparecen bordes no deseados provocados por la presencia de ruido, por el efecto de sombras de los objetos o provocados por una iluminación no uniforme en la escena.
Los contornos de una imagen suministran una valiosa información que es utilizada en muchas tareas de visión por computador, como la segmentación de la imagen o el reconocimiento de objetos, y que además reducen de manera significante la cantidad de información a manejar; eliminando aquella innecesaria pero preservando las características fundamentales de la imagen. Es por ello, que estos métodos son una herramienta muy poderosa en este campo.

Si se considera la intensidad de la imagen como una función, al buscar sus contornos, estaríamos buscando los picos de dicha función. En una función continua, el cálculo de la primera derivada nos sirve para encontrar sus picos. Si estamos en dos dimensiones, el cálculo de la primera derivada viene determinado por el gradiente. Las ecuaciones del gradiente se ven en la Figura C.1. En el caso de las imágenes, la intensidad de la imagen es una función discreta, por lo tanto, un aproximación al gradiente se basa en el cálculo de las diferencias entre los niveles de gris de la imagen, usando para ello máscara de convolución. La variación de gris se calcula como muestra la figura C.2.

Existen numerosos métodos de detección de contornos como son el LoG [13] o el explicado en [18], en este proyecto, para obtener los contornos de la imagen, se le aplica el método de Canny [19] por ser unos de los métodos más empleados.

Algoritmo de Canny El valor de la primera derivada es usado porque toma el valor cero en aquellas regiones donde no se produce un cambio de intensidad y tiene un valor constante en toda la transición de intensidad, por lo tanto, un cambio de intensidad se manifiesta como un cambio brusco en la primera derivada; característica usada para

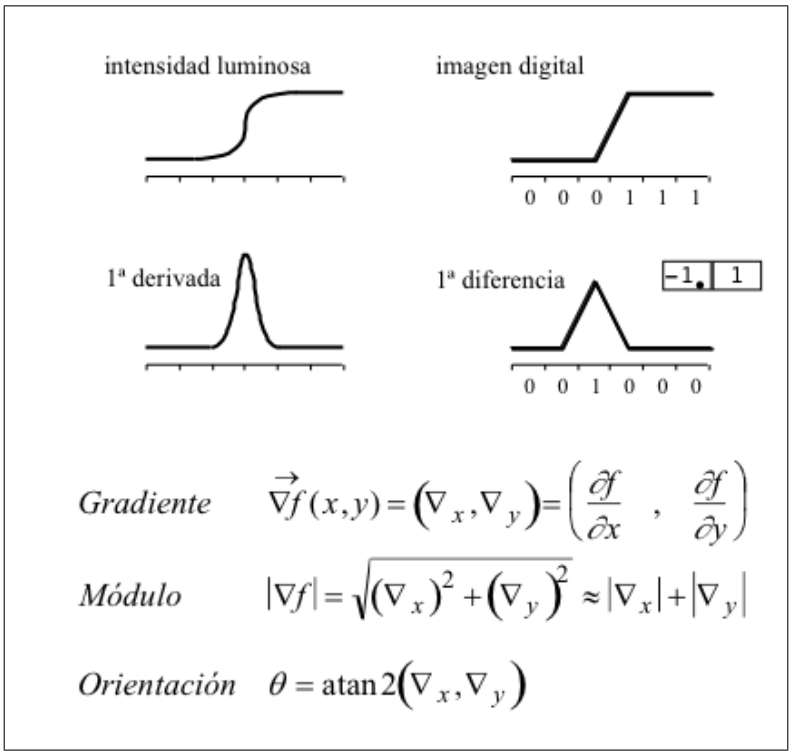


Figura C.1: Gradientes

$$K[m,n] = \begin{bmatrix} k_{-a,-c} & \cdots & k_{-a,0} & \cdots & k_{-a,d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ k_{0,-c} & \cdots & k_{0,0} & \cdots & k_{0,d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ k_{b,-c} & \cdots & k_{b,0} & \cdots & k_{b,d} \end{bmatrix}$$

$$g[i,j] = K[m,n] * f[i,j] = \sum_{m=-a}^{b} \sum_{n=-c}^{d} K[m,n] f[i+m, j+n]$$

Figura C.2: Máscara convolución

detectar un borde. El operador propuesto por Canny se aproxima mediante la derivada de la Gausiana en la dirección perpendicular al borde. Los pasos a seguir son los siguientes:

1. Calcular el módulo y la dirección del gradiente de la imagen suavizada aplicando un operdor DroG.

2. En la dirección del gradiente, eliminar puntos que no sean máximos locales del

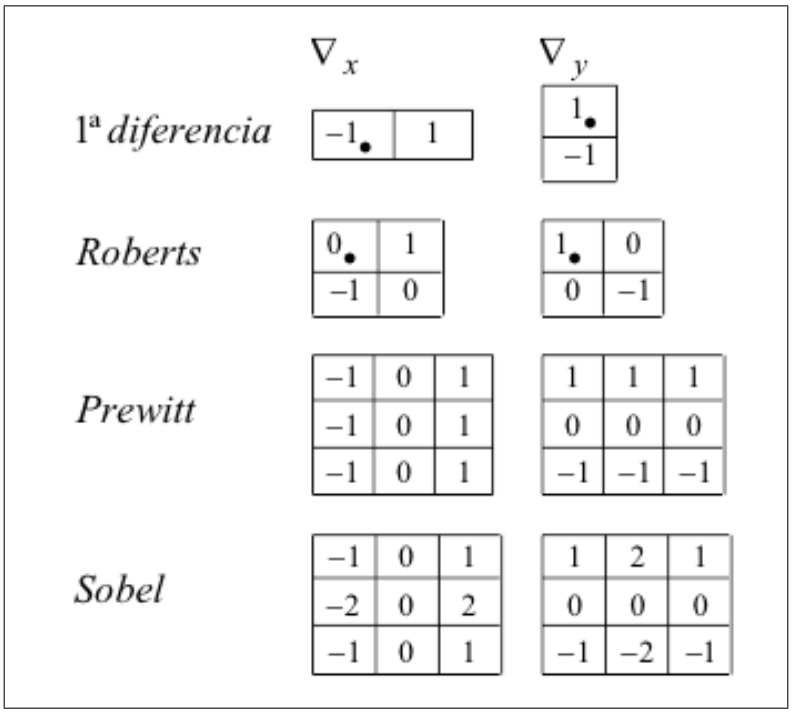


Figura C.3: Máscaras para el cálculo del gradiente

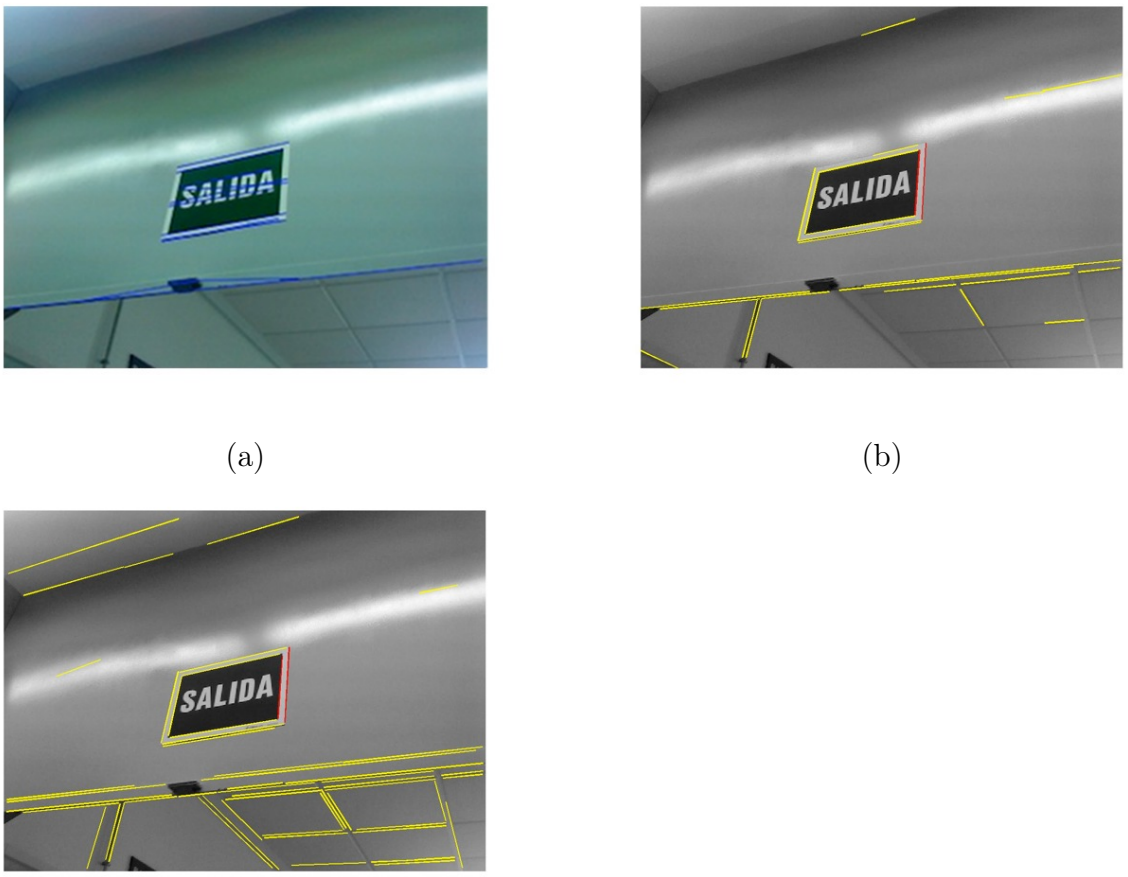
módulo para conseguir un adelgazamiento de los bordes hasta un píxel de ancho. Se consideran cuatro direcciones del gradiente 0º, 45º, 90º y 135º (con respecto al eje horizontal). Para cada píxel se encuentra aquella dirección que mejor se aproxime a su dirección del gradiente. Posteriormente se observa si el valor de la magnitud del gradiente es menor que sus vecinos en dicha dirección. De ser así, se le asigna un cero, en caso contrario, la magnitud del gradiente.

3. Se aplica una función de histéresis basada en dos umbrales: *un gradiente alto* y *un gradiente bajo*, que pretende reducir las falsas detecciones. Cada segmento de un borde debe tener un píxel cuyo módulo del gradiente supere el umbral máximo, así como todos los píxeles vecinos siguiendo la orientación del borde mientras que el valor de su gradiente no caiga por debajo del umbral mínimo.

C.1. Ejemplo de resultados proporcionados por varios extractores de rectas

En la Figura C.4 se ve un ejemplo de las rectas que se obtienen con los distintos métodos. Tanto el método de Burns como el finalmente implementado. Éstos proporcionan unos mejores resultados que el disponible en la librería de *OpenCV* y dado que había que adaptar cualquiera de ellos para poder utilizarlo en este proyecto, se decidió adaptar aquel

que proporcionaba mejores resultados.



(a) (b)

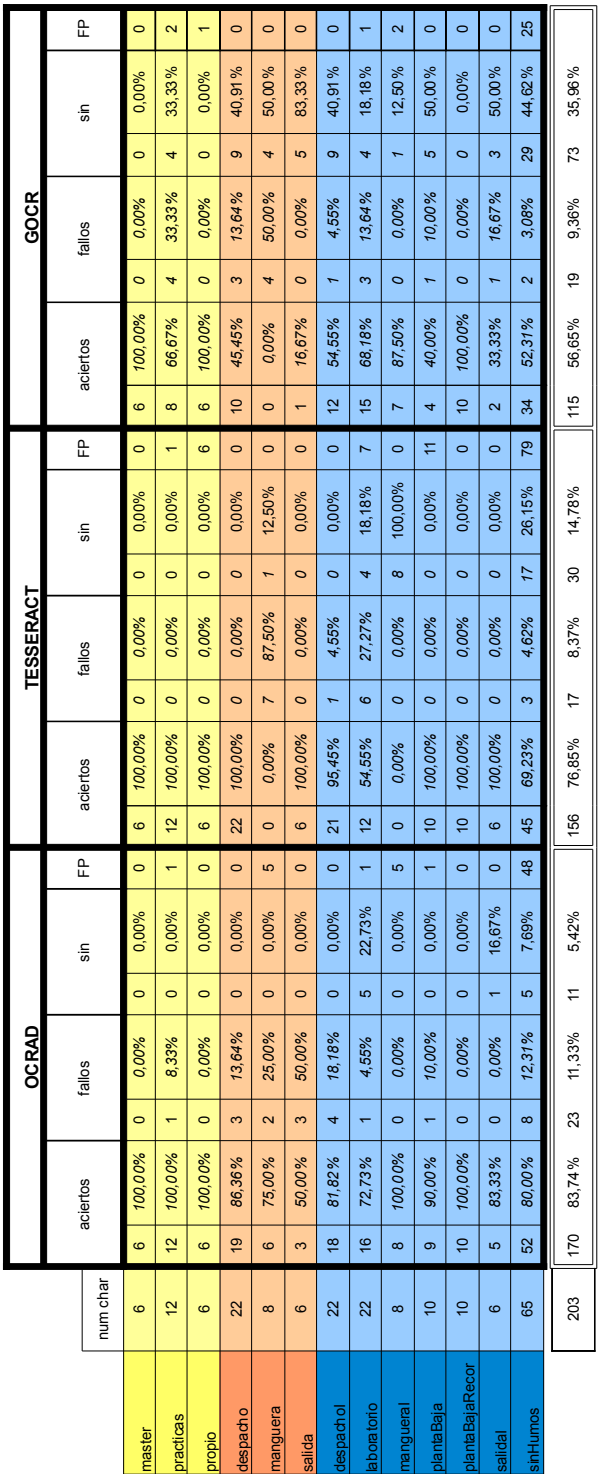
(c)

Figura C.4: Rectas obtenidas con distintos extractores: (a) Transformada de Hough proporcionada por la biblioteca OpenCV, (b) Método de Burns y (c) Extractor implementado para este proyecto

Apéndice D

Comparativa entre los OCRs disponibles

En este anexo se muestra los resultados obtenidos de comparar los tres *OCRs* de código disponible: *OCRAD*, *TESSERACT* y *GOCR*. Se han seleccionado varias imágenes, a partir de ellas, se ha recortado su texto, ya que los OCRs en la mayoría de los casos, no eran capaces de extraerlo por sí solos. Las imágenes también han sido binarizadas para facilitar el reconocimiento a los *OCRs*. En las comparaciones se ha tenido en cuenta la inclinación del texto, incluyendo imágenes inclinadas y estas mismas pero corrigiendo su inclinación manualmente. En la Figura D.1 se recogen los datos obtenidos, teniendo en cuenta el número de caracteres que se ha conseguido reconocer correctamente y aquellos que se han reconocido mal.
Tras la tabla, se muestran las imágenes de prueba y sus resultados con cada *OCR*, así como la binarización, recorte y rectificación de la inclinación según la imagen a analizar.



	num char	OCRAD						TESSERACT						GOCR					
		aciertos		fallos		sin	FP	aciertos		fallos		sin	FP	aciertos		fallos		sin	FP
master	6	6	100,00%	0	0,00%	0	0	6	100,00%	0	0,00%	0	0	6	100,00%	0	0,00%	0	0
practicas	12	12	100,00%	1	8,33%	0	1	12	100,00%	0	0,00%	0	1	8	66,67%	4	33,33%	4	2
propio	6	6	100,00%	0	0,00%	0	0	6	100,00%	0	0,00%	0	6	6	100,00%	0	0,00%	0	1
despacho	22	19	86,36%	3	13,64%	0	0	22	100,00%	0	0,00%	0	0	10	45,45%	3	13,64%	9	0
manguera	8	6	75,00%	2	25,00%	0	5	0	0,00%	7	87,50%	1	0	0	0,00%	4	50,00%	4	0
salida	6	3	50,00%	3	50,00%	0	0	6	100,00%	0	0,00%	0	0	1	16,67%	0	0,00%	5	0
despacho1	22	18	81,82%	4	18,18%	0	0	21	95,45%	1	4,55%	0	0	12	54,55%	1	4,55%	9	0
laboratorio	22	16	72,73%	1	4,55%	5	1	12	54,55%	6	27,27%	4	7	15	68,18%	3	13,64%	4	1
mangueral	8	8	100,00%	0	0,00%	0	5	0	0,00%	0	0,00%	8	0	7	87,50%	0	0,00%	1	2
plantaBaja	10	9	90,00%	1	10,00%	0	1	10	100,00%	0	0,00%	0	11	4	40,00%	1	10,00%	5	0
plantaBajaRecor	10	10	100,00%	0	0,00%	0	0	10	100,00%	0	0,00%	0	0	10	100,00%	0	0,00%	0	0
salidal	6	5	83,33%	0	0,00%	1	0	6	100,00%	0	0,00%	0	0	2	33,33%	1	16,67%	3	0
sinHumos	65	52	80,00%	8	12,31%	5	48	45	69,23%	3	4,62%	17	79	34	52,31%	2	3,08%	29	25
	203	170	83,74%	23	11,33%	11	5,42%	156	76,85%	17	8,37%	30	14,78%	115	56,65%	19	9,36%	73	35,96%

Figura D.1: Resultados de la comparación entre los tres OCRs

D. Comparativa entre los OCRs disponibles

OCRAD

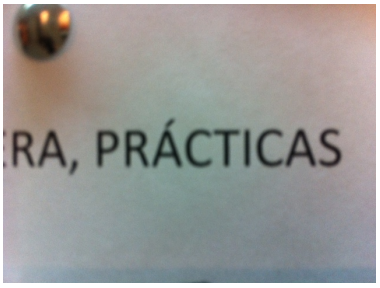
```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/masterBinario.pbm
MASTER
```

TESSERACT

```
robot21:Desktop anabc$ more resul.txt
MASTER
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/masterBinario.pbm
MASTER
```



OCRAD

```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/practicasBinaria.pbm
_
RA, PRACTICAS
```

TESSERACT

```
robot21:Desktop anabc$ tesseract fotos_prueba_OCR/jpg/practicasBinaria.tif resul -l spa
robot21:Desktop anabc$ more resul.txt
G
RA, PRÁCTICAS
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/practicasBinaria.pbm
_
;Rn, _RAcTlcns
```

OCRAD

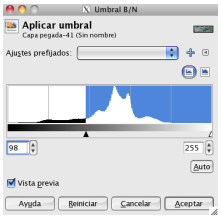
```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/propioBinaria.pbm
propio
```

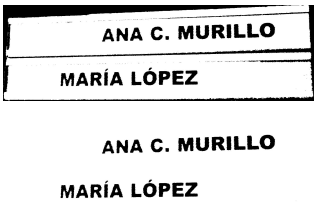
TESSERACT

```
robot21:Desktop anabc$ tesseract fotos_prueba_OCR/jpg/propioBinaria.tif resul -l spa
robot21:Desktop anabc$ more resul.txt
blIVdb
propio
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/propioBinaria.pbm
_
pro pio
```





OCRAD

```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/despachoBinariaRecorte.pbm
AHa c. MURILLO
MAR?A L?PEZ
```

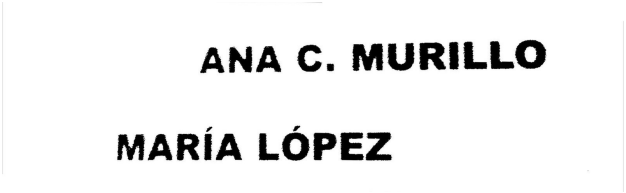
TESSERACT

```
robot21:Desktop anabc$ more resul.txt
ANA c. MURILLO
MARiA LÓPEZ
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/despachoBinariaRecorte.pbm
__ ___r A_ Al0f_p_ Em2gl _ _ _ _0
```

D. Comparativa entre los OCRs disponibles



OCRAD

```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/despachoBinariaRecorteI.pbm
AHa c, MURILLO
MAR?A L?PEZ
```

TESSERACT

```
robot21:Desktop anabc$ more resul.txt
ANA c. MURILLO
MAR¡A LÓPEZ
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/despachoBinariaRecorteI.pbm
____rA l__ p Eg
__A C. m_RI__0
```

OCRAD

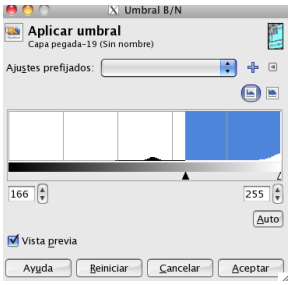
```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/laboratorioRecortada.pbm
_BO_TORIO
&OMECANICA
e__
```

TESSERACT

```
robot21:Desktop anabc$ tesseract fotos_prueba_OCR/jpg/laboratorioRecortada.tif resul -l spaxt L
›\somom0
WOMECANICA
ëla=_¡.¡_
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/laboratorioRecortada.pbm
_eoRATORlO
_omECnNICA
__x _
```



OCRAD

```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/mangueraRecortada.pbm
llllL_ _
MpNGUERp
```

TESSERACT

```
robot21:Desktop anabc$ tesseract fotos_prueba_OCR/jpg/mangueraRecortada.tif resul -l spa
robot21:Desktop anabc$ more resul.txt
QIIIII"
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/mangueraRecortada.pbm
__<barcode type="unknown" />
_r__ lll
```

OCRAD

```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/mangueraRecortadaI.pbm
llllL_ _
-
-
MANGUERA
```

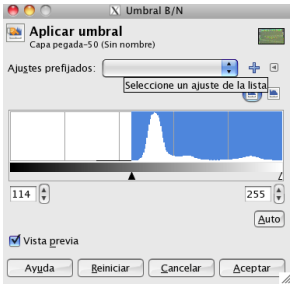
TESSERACT

```
robot21:Desktop anabc$ tesseract fotos_prueba_OCR/jpg/mangueraRecortadaI.tif resul -l spa
Tesseract Open Source OCR Engine
Image has 8 * 4 bits per pixel, and size (398,712)
Resolution=72
Empty page
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/mangueraRecortadaI.pbm
__<barcode type="unknown" />
__
'   _
-_
M_NGUERA
```

D. Comparativa entre los OCRs disponibles



OCRAD

```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/plantaBajaBinarizada.pbm
PLANTA BAJD^
```

TESSERACT

```
robot21:Desktop anabc$ tesseract fotos_prueba_OCR/jpg/plantaBajaBinarizada.tif resul -l spa
Tesseract Open Source OCR Engine
Image has 8 * 4 bits per pixel, and size (1120,670)
Resolution=72
robot21:Desktop anabc$ more resul.txt PLANTA  B'AJA
.~ l~~~>¬.~
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/plantaBajaBinarizada.pbm __
_LANT_ _n___
____
```

Imagen recorta, invertida y umbralizada. Borrar los bordes.

OCRAD

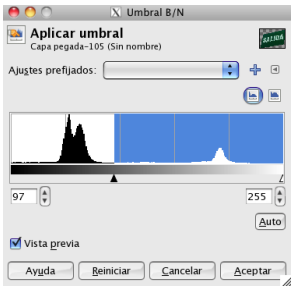
```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/plantaBajaRecortada.pbm
PLANTA BAJA
```

TESSERACT

```
robot21:Desktop anabc$ tesseract fotos_prueba_OCR/jpg/plantaBajaRecortada.tif resul -l spa
robot21:Desktop anabc$ more resul.txt
PLANTA BAJA
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/plantaBajaRecortada.pbm
PLANTA BAJA
```



OCRAD

```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/salidaBinarizada.pbm
SOLIDD
```

TESSERACT

```
robot21:Desktop anabc$ tesseract fotos_prueba_OCR/jpg/salidaBinarizada.tif resul -l spa
Tesseract Open Source OCR Engine
robot21:Desktop anabc$ more resul.txt
SALIDA
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/salidaBinarizada.pbm
s_____
```

OCRAD

```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/salidaBinarizadaI.pbm
SA_IDA
```

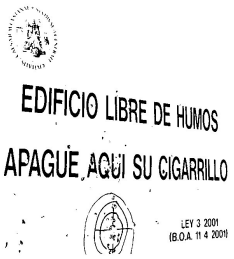
TESSERACT

```
robot21:Desktop anabc$ tesseract fotos_prueba_OCR/jpg/salidaBinarizadaI.tif resul -l spa
Tesseract Open Source OCR Engine
Image has 8 * 4 bits per pixel, and size (483,357)
Resolution=72
robot21:Desktop anabc$ more resul.txt
SALIDA
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/salidaBinarizadaI.pbm
S_ii__
```

D. Comparativa entre los OCRs disponibles



OCRAD

```
robot21:Desktop anabc$ ocrad fotos_prueba_OCR/jpg/sinHumosBinarizada.pbm ,\\\ _ ,,
_ , ' "'r
\Wt,
_L?_'`__ _
|
r, -7_ _

__?FICIO L?BRE DE HuMo_

APAf'U'E_qC,_|. SU C,|.GARRILLO

. |
?
_..(. .^ \ .. iEY 3 _001
' . '_? IB.O.A. 11 4 IO01|
l \
```

TESSERACT

```
robot21:Desktop anabc$ tesseract fotos_prueba_OCR/jpg/sinHumosBinarizada.tif resul -l spa
Tesseract Open Source OCR Engine
Image has 8 * 4 bits per pixel, and size (625,736)
Resolution=72
robot21:Desktop anabc$ more resul.txt \,
.\`\\ M@
` \ *R 'b
2 ,f 23 ,
1 *fäh *
LL › Ñ
"'Ium¬
EDIFICIO VLÍBRE DE Humos
APAGUEAGUI' SU GIGARRILLO
`_*   ~
ƒ- _     (s.o.A,
```

GOCR

```
robot21:Desktop anabc$ gocr -m 4 fotos_prueba_OCR/jpg/sinHumosBinarizada.pbm
_E_l_  __t_rt   l  >
__:
_
J   l
'_ _ _=
_( _
I  _t
_
_IIrt_(___
D<barcode type="unknown" /><barcode type="unknown" />
F   01_'B_E _ E H' u m o s
A_AGUE,___G.ISU<barcode type="unknown" /><barcode type="unknown" />
<barcode type="unknown" /><barcode type="unknown" />
G    O
_
_         _ J _ t
'
/    ( B. o. A.
      11 4 2 O O
```

Apéndice E

Diagramas de navegación

El funcionamiento general de la aplicación se resumen en tres pasos como muestra el diagrama diagrama E.1:

1. El usuario debe seleccionar una imagen
2. De manera transparente al usuario, se debe calcular las hipótesis de carteles
3. Se muestran los resultados de leer con cada hipótesis aceptada con un OCR.

La imagen puede elegirse, como puede verse en el diagrama E.2, tanto de las almacenadas en la propia biblioteca del dispositivo como capturando una nueva con la cámara, siempre que el dispositivo disponga de ella.

El proceso realizado y detallado en el Capítulo 2 para extraer las hipótesis de los rectángulos se realiza de una manera transparente al usuario, pero para comprobar el funcionamiento de la implementación y posterior depuración de la misma, se ha realizado una navegación secundaria que permite realizar dicho proceso paso a paso. El diagrama E.3 permite realizar las tareas especificadas en la sección 2.1. El diagrama E.4 muestra la navegación para llevar a cabo la evaluación de una hipótesis tal y como se explica en la sección 2.2. En el diagrama E.5 se observa el proceso detallado en la sección 2.3 y 2.4.

El último paso de la aplicación es donde se muestran al usuario los resultados obtenidos con las hipótesis aceptadas. La tarea individual de leer cada hipótesis también se puede realizar de manera individual gracias a la navegación, puede verse en el diagrama E.6.

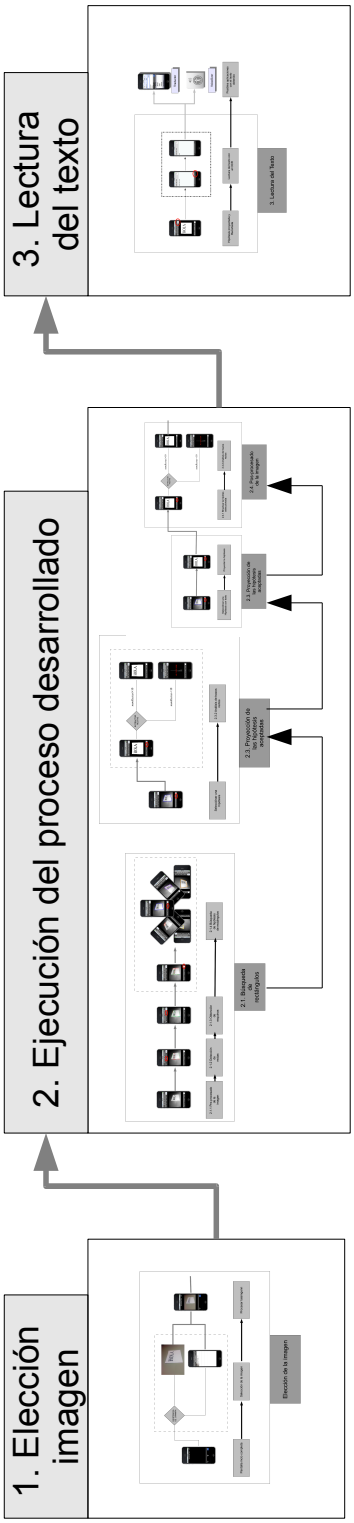


Figura E.1: Funcionamiento general de la aplicación

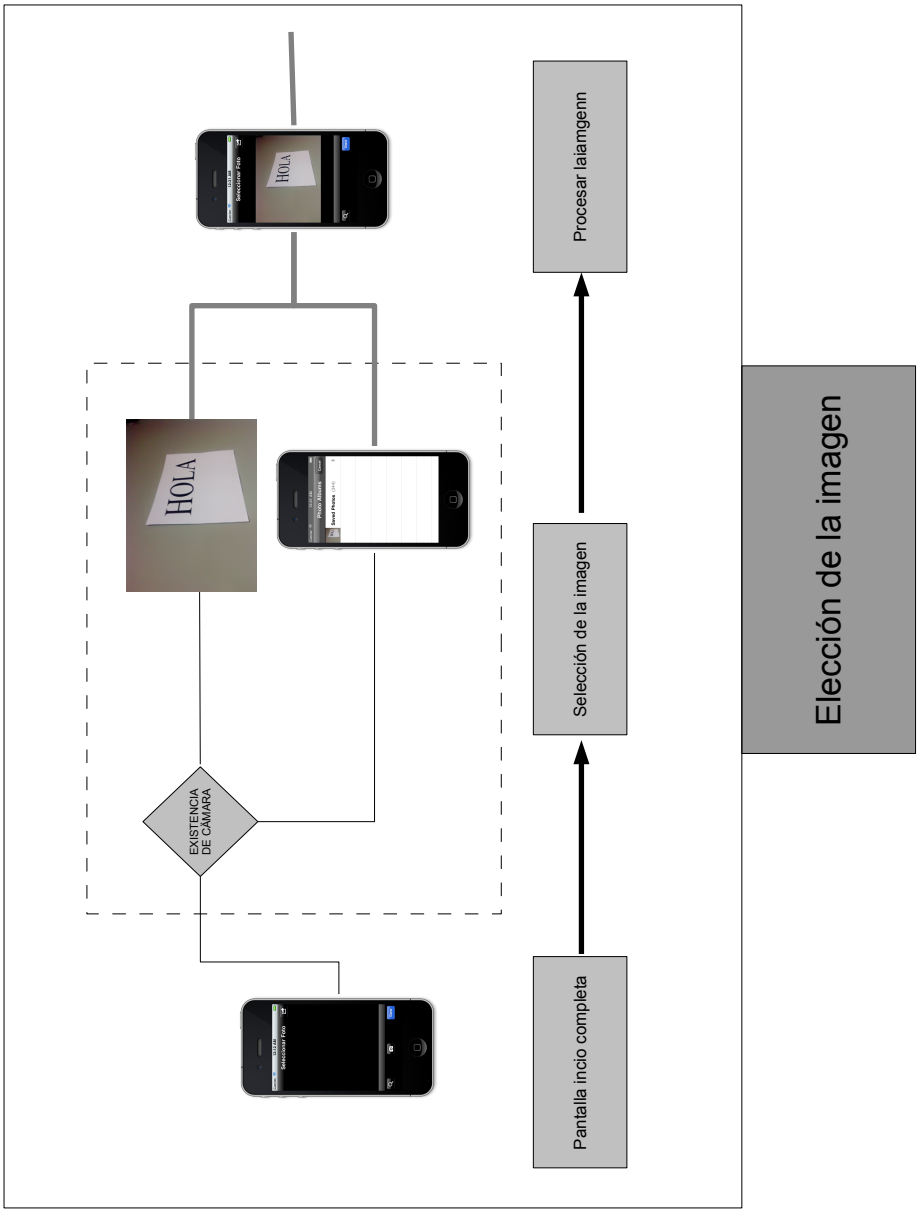


Figura E.2: Elección de la imagen a procesar

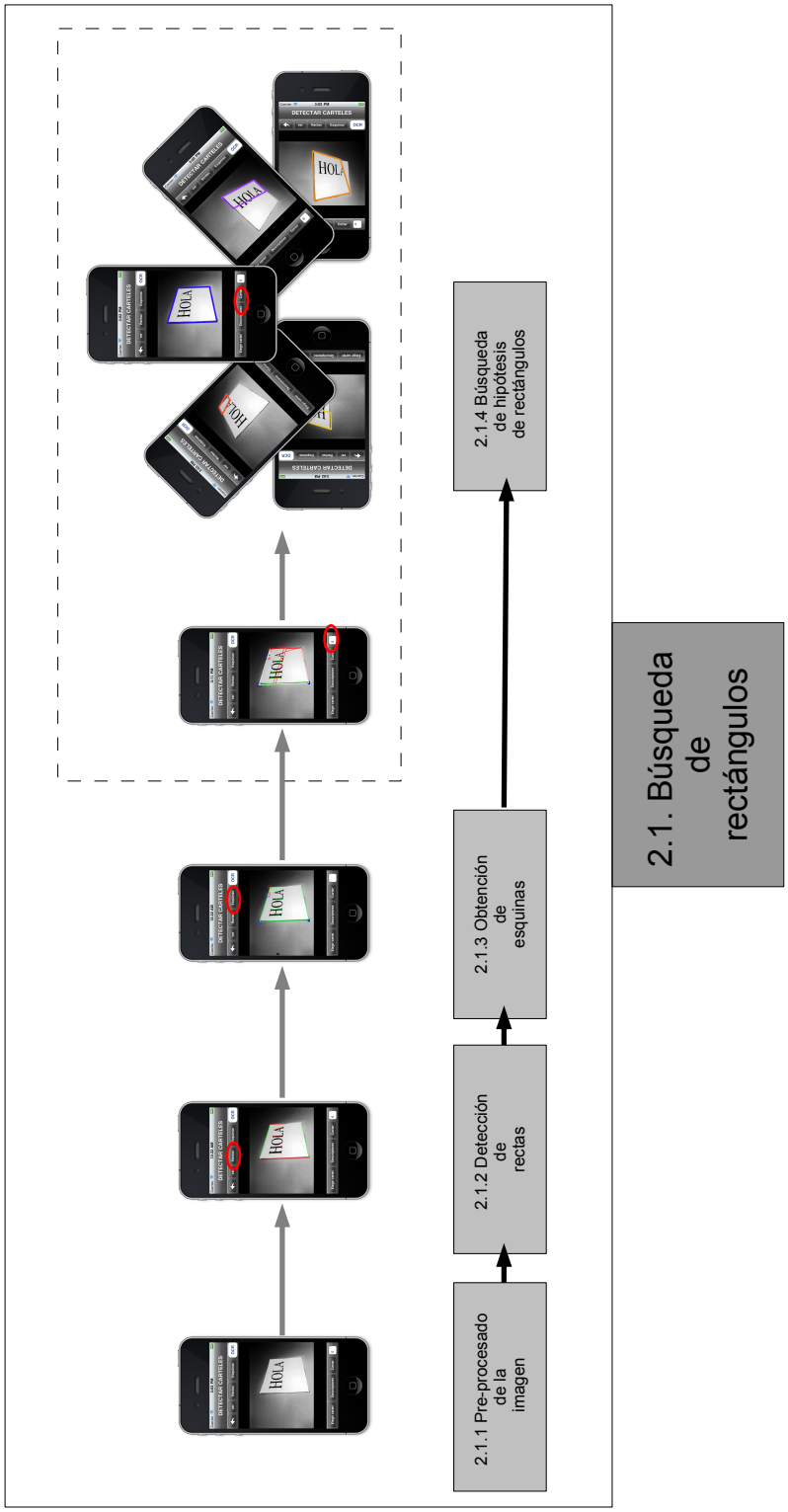


Figura E.3: Búsqueda de rectángulos

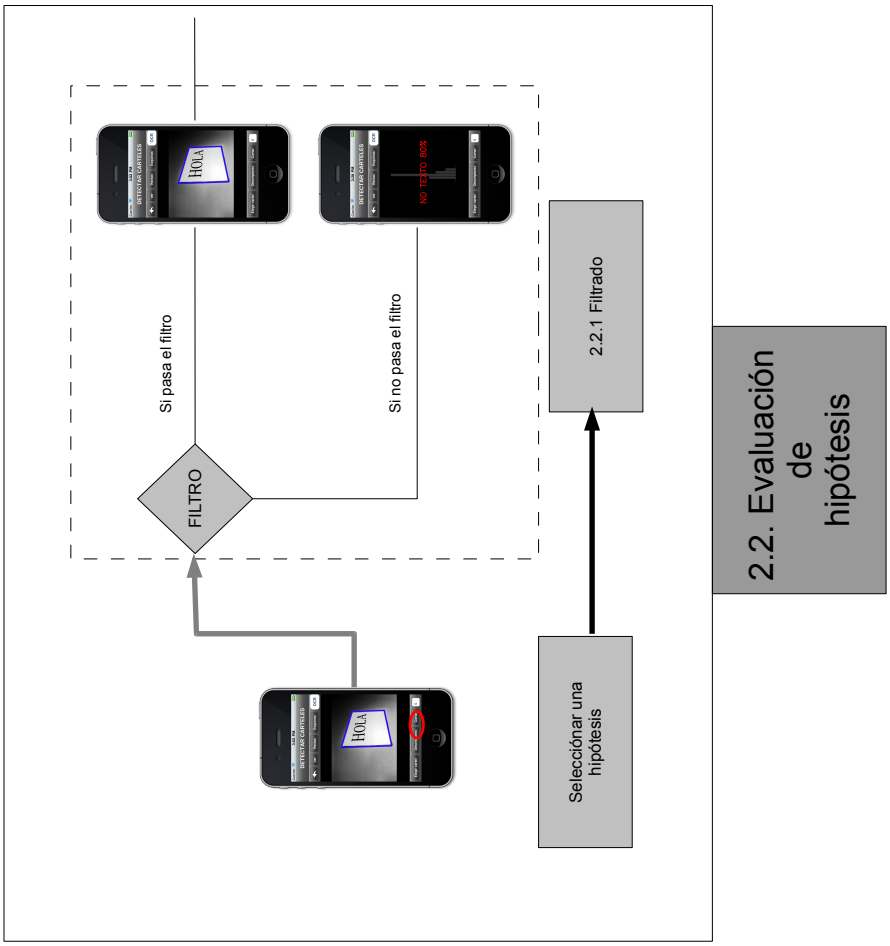


Figura E.4: Evaluación de hipótesis

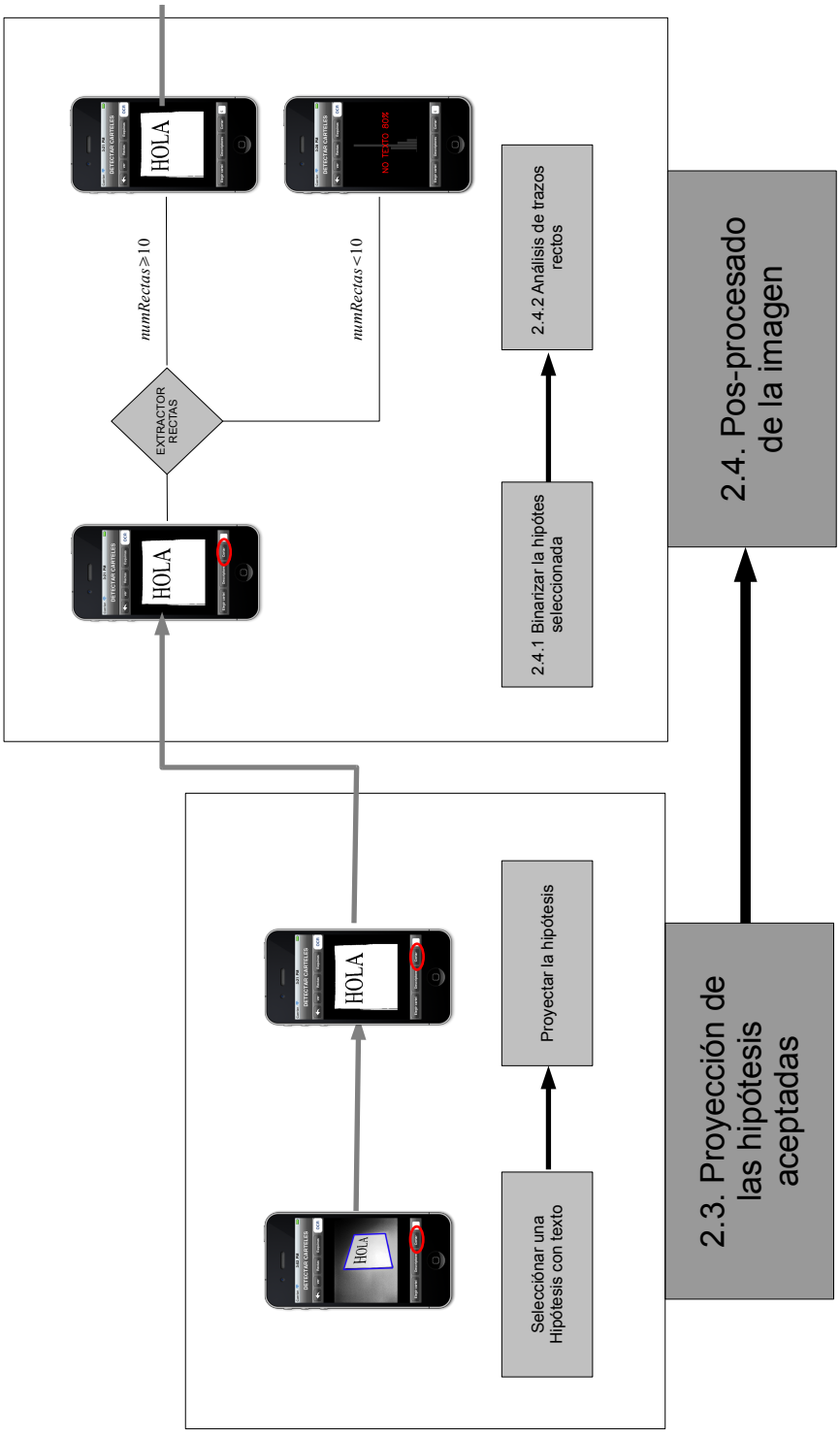


Figura E.5: Proyección y posprocesado de las hipótesis

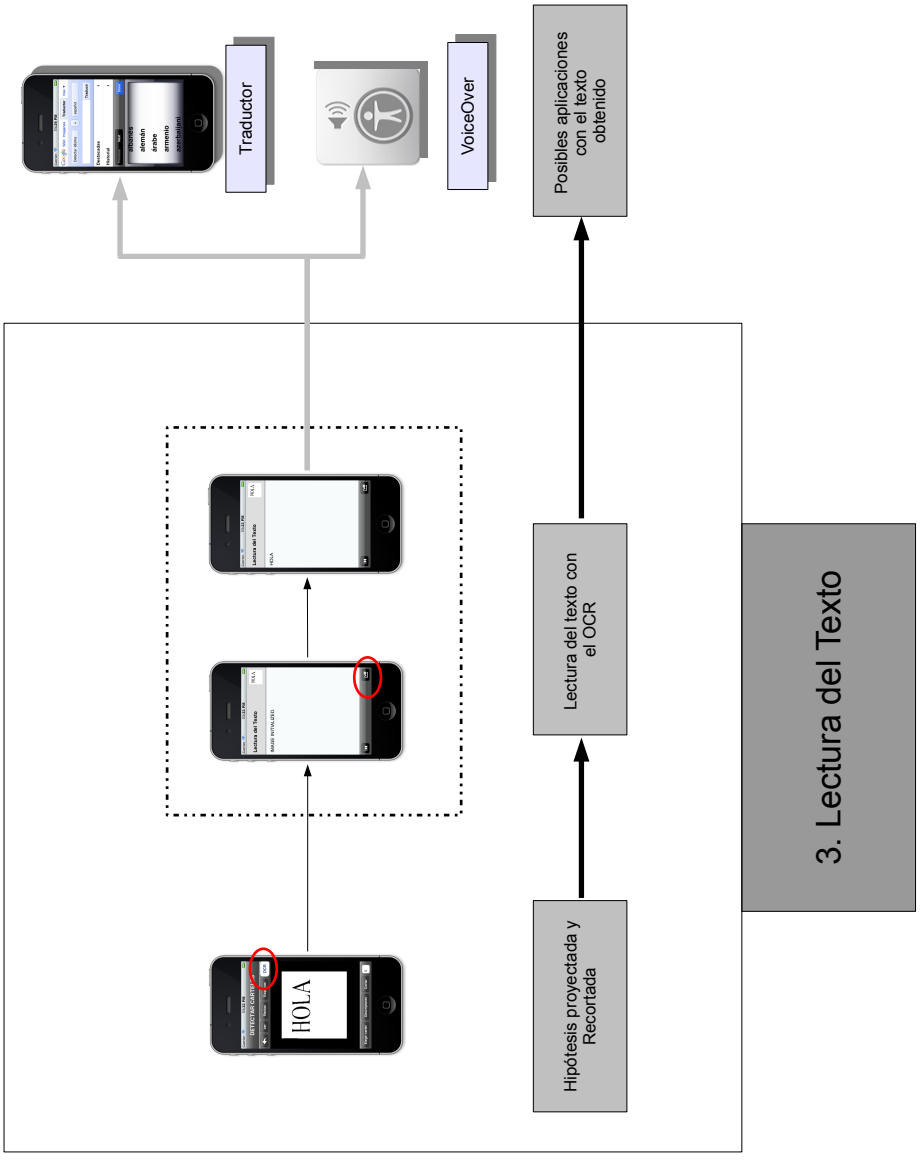


Figura E.6: Lectura del texto

Apéndice F

Resultados empleados en la comparación

En este anexo se adjuntan las tablas con los datos empleados para realizar la comparación entre la aplicación desarrollada y el OCR así como las imágenes empleadas para tal comparación.

PFC	PLANTA BAJA		PULSADOR DE ALARMA		TERCERO		L.003 LABORATORIO		LEY 3/2001 B.O.A. 11/4/2001		PROHIBIDO FUMAR EN ESTE EDIFICIO	
HIPOTESIS		%		%		%		%		%		%
TOTAL	*40*		*118*		*1*		*58*		*18*		*18*	
No proyectar	18	45,0%	12	10,2%	1	100,0%	16	27,6%	4	22,2%	4	22,2%
No trazos rectos	21	52,5%	67	56,8%	0	0,0%	13	22,4%	1	5,6%	1	5,6%
ACEPTADAS	1	2,5%	39	33,1%	0	0,0%	29	50,0%	13	72,2%	13	72,2%
H buenas	1	100,0%	23	59,0%	1	100,0%	24	82,8%	3	23,1%	3	23,1%
CARACTERES		%		%		%		%		%		%
TOTAL	10		16		7		16		24		28	
aciertos	10	100,0%	16	100,0%	6	85,7%	15	93,8%	18	75,0%	27	96,4%
fallos	0	0,0%	0	0,0%	1	14,3%	1	6,3%	5	20,8%	1	3,6%
ruido	15	150,0%	0	0,0%	1	14,3%	0	0,0%	0	0,0%	0	0,0%
LINEAS		%		%		%		%		%		%
TOTAL	1		2		1		1		2		3	
aciertos	1	100,0%	2	100,0%	1	100,0%	1	100,0%	2	100,0%	3	100,0%
ruido	1	100,0%	0	0,0%	0	0,0%	0	0,0%	0	0,0%	0	0,0%

OCR	PLANTA BAJA		PULSADOR DE ALARMA		TERCERO		L.003 LABORATORIO		LEY 3/2001 B.O.A. 11/4/2001		PROHIBIDO FUMAR EN ESTE EDIFICIO	
CARACTERES		%		%		%		%		%		%
TOTAL	10		16		7		16		24		28	
aciertos	10	100,0%	0	0,0%	4	57,1%	0	0,0%	8	33,3%	24	85,7%
fallos	0	0,0%	0	0,0%	2	28,6%	0	0,0%	8	33,3%	4	14,3%
ruido	32	320,0%	0	0,0%	1	14,3%	0	0,0%	6	25,0%	4	14,3%
LINEAS		%		%		%		%		%		%
TOTAL	1		2		1		1		2		3	
aciertos	1	100,0%	0	0,0%	1	100,0%	0	0,0%	2	100,0%	3	100,0%
ruido	9	900,0%	30	1500,0%	0	0,0%	20	2000,0%	1	50,0%	0	0,0%

Figura F.1: Tablas para la comparación de imágenes frontales

PFC	INFRAESTRUCTURA		EXTINTOR		ASEOS WC (BLANCO)		YOGA SALUD INTEGRAL entlo. Izqa.		JESUS MURILLO RAMON PETRA Mº ARNAL GIL 3ºB	
HIPOTESIS		%		%		%		%		%
TOTAL	*6*		*27*		*20*		*19*		*8*	
No proyectar	2	33,3%	8	29,6%	4	20,0%	3	15,8%	0	0,0%
No trazos rectos	0	0,0%	12	44,4%	8	40,0%	11	57,9%	0	0,0%
ACEPTADAS	4	66,7%	7	25,9%	8	40,0%	5	26,3%	8	100,0%
H buenas	4	100,0%	7	100,0%	8	100,0%	5	100,0%	8	100,0%
CARACTERES		%		%		%		%		%
TOTAL	15		8		7		29		35	
aciertos	15	100,0%	8	100,0%	3	42,9%	14	48,3%	33	94,3%
fallos	0	0,0%	0	0,0%	5	71,4%	5	17,2%	2	5,7%
ruido	0	0,0%	0	0,0%	4	57,1%	0	0,0%	0	0,0%
LINEAS		%		%		%		%		%
TOTAL	1		1		1		3		3	
aciertos	1	100,0%	1	100,0%	1	100,0%	2	66,7%	3	100,0%
ruido	0	0,0%	0	0,0%	0	0,0%	0	0,0%	0	0,0%

OCR	INFRAESTRUCTURA		EXTINTOR		ASEOS WC (BLANCO)		YOGA SALUD INTEGRAL entlo. Izqa.		JESUS MURILLO RAMON PETRA Mº ARNAL GIL 3ºB	
CARACTERES		%		%		%		%		%
TOTAL	15		8		7		29		35	
aciertos	0	0,0%	8	100,0%	7	100,0%	19	65,5%	15	42,9%
fallos	0	0,0%	0	0,0%	0	0,0%	3	10,3%	14	40,0%
ruido	0	0,0%	0	0,0%	25	357,1%	40	137,9%	0	0,0%
LINEAS		%		%		%		%		%
TOTAL	1		1		1		3		3	
aciertos	0	0,0%	1	100,0%	1	100,0%	3	100,0%	3	100,0%
ruido	0	0,0%	0	0,0%	6	600,0%	8	266,7%	0	0,0%

Figura F.2: Tablas para la comparación de imágenes frontales

PFC	ZARAGOZA PLAZA AEROPUERTO		CORONAS RAMOS DE NOVIA ADORNOS FLORALES 31 94 55 37 42 35		ASEOS WC (AZUL)		SEMINARIOS...		SANTANDER	
HIPOTESIS		%		%		%		%		%
TOTAL	*94*		*17*		*37*		*7*		*5*	
No proyectar	34	36,2%	3	17,6%	16	43,2%	3	42,9%	1	20,0%
No trazos rectos	25	26,6%	4	23,5%	17	45,9%	0	0,0%	2	40,0%
ACEPTADAS	35	37,2%	10	58,8%	4	10,8%	4	57,1%	2	40,0%
H buenas	18	51,4%	9	90,0%	1	25,0%	4	100,0%	1	50,0%
CARACTERES		%		%		%		%		%
TOTAL	23		46		6		74		9	
aciertos	15	65,2%	46	100,0%	5	83,3%	0	0,0%	4	44,4%
fallos	3	13,0%	0	0,0%	1	16,7%	0	0,0%	10	111,1%
ruido	0	0,0%	2	4,3%	3	50,0%	0	0,0%	0	0,0%
LINEAS		%		%		%		%		%
TOTAL	3		5		1		4		1	
aciertos	3	100,0%	5	100,0%	1	100,0%	0	0,0%	1	100,0%
ruido	0	0,0%	0	0,0%	1	100,0%	1	25,0%	0	0,0%

OCR	ZARAGOZA PLAZA AEROPUERTO		FLORES RAMOS DE NOVIA ADORNOS FLORALES 31 94 55 37 42 35		ASEOS WC (AZUL)		SEMINARIOS...		SANTANDER	
CARACTERES		%		%		%		%		%
TOTAL	23		46		6		74		9	
aciertos	0	0,0%	5	10,9%	0	0,0%	26	35,1%	0	0,0%
fallos	0	0,0%	0	0,0%	0	0,0%	29	39,2%	0	0,0%
ruido	70	304,3%	45	97,8%	58	966,7%	8	10,8%	200	2222,2%
LINEAS		%		%		%		%		%
TOTAL	3		5		1		4		1	
aciertos	0	0,0%	1	20,0%	0	0,0%	4	100,0%	0	0,0%
ruido	19	633,3%	8	160,0%	21	2100,0%	1	25,0%	30	3000,0%

Figura F.3: Tablas para la comparación de imágenes frontales

Figura F.4: Las 8 primeras Imágenes de prueba frontales

Figura F.5: Las 8 últimas Imágenes de prueba frontales

F. Resultados empleados en la comparación

PFC

HIPOTESIS	HOLA		SOLO ENVASES DE VIDRIO		EDIFICIO LIBRE DE HUMOS APAGUE aquí SU CIGARRILLO		ACCESO A PLANTAS(lejos)		SALIDA (lejos)	
		%		%		%		%		%
TOTAL	7		30		17		19		13	
No proyectar	1	14,3%	14	46,7%	4	23,5%	7	36,8%	6	46,2%
No trazos rectos	0	0,0%	3	10,0%	2	11,8%	9	47,4%	4	30,8%
ACEPTADAS	6	85,7%	13	43,3%	11	64,7%	3	15,8%	3	23,1%
H buenas	6	100,0%	13	100,0%	11	64,7%	2	66,7%	1	33,3%
CARACTERES		%		%		%		%		%
TOTAL	4		19		42		14		6	
aciertos	4	100,0%	11	57,9%	35	83,3%	8	57,1%	6	100,0%
fallos	0	0,0%	0	0,0%	4	9,5%	6	42,9%	0	0,0%
ruido	0	0,0%	9	47,4%	9	21,4%	3	21,4%	1	16,7%
LINEAS		%		%		%		%		%
TOTAL	1		1		2		1		1	
aciertos	1	100,0%	1	100,0%	2	100,0%	1	100,0%	1	100,0%
ruido	0	0,0%	2	200,0%	2	100,0%	2	200,0%	0	0,0%

OCR

CARACTERES	HOLA		SOLO ENVASES DE VIDRIO		EDIFICIO LIBRE DE HUMOS APAGUE aquí SU CIGARRILLO		ACCESO A PLANTAS(lejos)		SALIDA (lejos)	
		%		%		%		%		%
TOTAL	4		19		42		14		6	
aciertos	0	0,0%	8	42,1%	0	0,0%	3	21,4%	0	0,0%
fallos	0	0,0%	0	0,0%	0	0,0%	12	85,7%	0	0,0%
ruido	0	0,0%	7	36,8%	0	0,0%	19	135,7%	0	0,0%
LINEAS		%		%		%		%		%
TOTAL	1		1		2		1		1	
aciertos	0	0,0%	1	100,0%	0	0,0%	1	100,0%	0	0,0%
ruido	6	600,0%	3	300,0%	9	450,0%	4	400,0%	35	3500,0%

Figura F.6: Tablas para la comparación de imágenes NO frontales

PFC

HIPOTESIS	NATIONAL PARK		DIRECCION		SEPTIMO		MANGUERA		ACCESO A PLANTAS(cerca)	
		%		%		%		%		%
TOTAL	2		21		4		56		4	
No proyectar	0	0,0%	17	81,0%	3	75,0%	25	44,6%	2	50,0%
No trazos rectos	0	0,0%	4	19,0%	1	25,0%	16	28,6%	1	25,0%
ACEPTADAS	2	100,0%	0	0,0%	0	0,0%	15	26,8%	1	25,0%
H buenas	2	100,0%	0	0,0%	0	0,0%	3	20,0%	1	100,0%
CARACTERES		%		%		%		%		%
TOTAL	13		23		7		1		14	
aciertos	7	53,8%	0	0,0%	0	0,0%	0	0,0%	10	71,4%
fallos	5	38,5%	0	0,0%	0	0,0%	0	0,0%	1	7,1%
ruido	28	215,4%	0	0,0%	0	0,0%	0	0,0%	4	28,6%
LINEAS		%		%		%		%		%
TOTAL	1		2		1		1		1	
aciertos	1	100,0%	0	0,0%	0	0,0%	0	0,0%	1	100,0%
ruido	3	300,0%	0	0,0%	0	0,0%	0	0,0%	0	0,0%

OCR

CARACTERES	NATIONAL PARK		DIRECCION		SEPTIMO		MANGUERA		ACCESO A PLANTAS(cerca)	
		%		%		%		%		%
TOTAL	13		23		7		1		14	
aciertos	7	53,8%	0	0,0%	0	0,0%	0	0,0%	0	0,0%
fallos	7	53,8%	0	0,0%	0	0,0%	0	0,0%	0	0,0%
ruido	40	307,7%	42	182,6%	80	1142,9%	31	3100,0%	60	428,6%
LINEAS		%		%		%		%		%
TOTAL	1		2		1		1		1	
aciertos	1	100,0%	0	0,0%	0	0,0%	0	0,0%	0	0,0%
ruido	4	400,0%	4	200,0%	24	2400,0%	5	500,0%	12	1200,0%

Figura F.7: Tablas para la comparación de imágenes NO frontales

PFC	SALIDA(cerca)		ANA C. MURRILLO MARIA LOPEZ		PLANTA BAJA (seccion)		PLANTA BAJA (pequeña)	
HIPOTESIS		%		%		%		%
TOTAL	*18*		*43*		*8*		*13*	
No proyectar	1	5,6%	16	37,2%	3	37,5%	6	46,2%
No trazos rectos	14	77,8%	13	30,2%	5	62,5%	4	30,8%
ACEPTADAS	3	16,7%	14	32,6%	0	0,0%	3	23,1%
H buenas	3	100,0%	6	42,9%	0	0,0%	0	0,0%
CARACTERES		%		%		%		%
TOTAL	6		22		10		10	
aciertos	0	0,0%	6	27,3%	0	0,0%	0	0,0%
fallos	0	0,0%	0	0,0%	0	0,0%	0	0,0%
ruido	0	0,0%	2	9,1%	0	0,0%	0	0,0%
LINEAS		%		%		%		%
TOTAL	1		2		1		1	
aciertos	1	100,0%	1	50,0%	0	0,0%	0	0,0%
ruido	0	0,0%	0	0,0%	0	0,0%	0	0,0%

OCR	SALIDA(cerca)		ANA C. MURRILLO MARIA LOPEZ		PLANTA BAJA (seccion)		PLANTA BAJA (pequeña)	
CARACTERES		%		%		%		%
TOTAL	6		22		10		10	
aciertos	0	0,0%	2	9,1%	0	0,0%	0	0,0%
fallos	0	0,0%	21	95,5%	0	0,0%	0	0,0%
ruido	11	183,3%	4	18,2%	52	520,0%	62	620,0%
LINEAS		%		%		%		%
TOTAL	1		2		1		1	
aciertos	0	0,0%	2	100,0%	0	0,0%	0	0,0%
ruido	1	100,0%	1	50,0%	17	1700,0%	13	1300,0%

Figura F.8: Tablas para la comparación de imágenes NO frontales

Figura F.9: Las 8 primeras Imágenes de prueba NO frontales

Figura F.10: Las 6 últimas Imágenes de prueba NO frontales